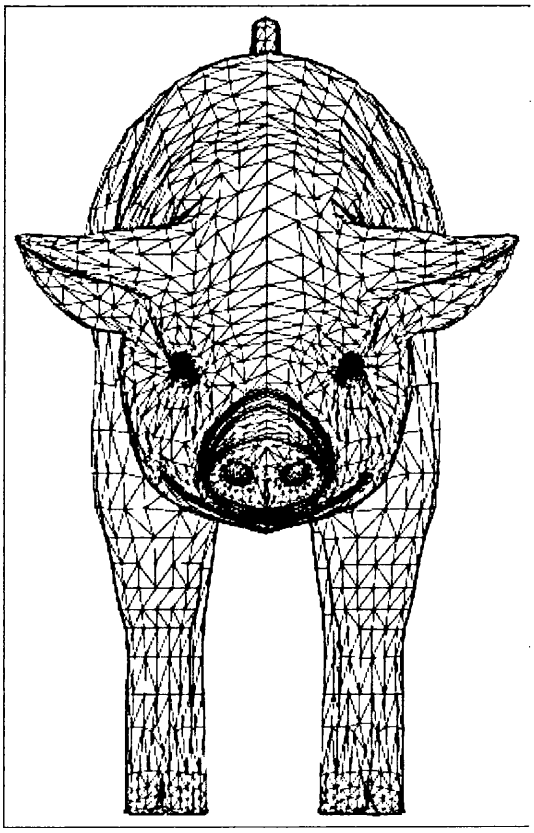
Unstructured mesh generation for mesh improvement techniques and contour meshing.

Bushrod, Rebecca

Unstructured Mesh Generation For Mesh Improvement Techniques And Contour Meshing



By

Rebecca Bushrod

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed.....(Candidate)
Date.....15/9/05.....

Statement 1

This thesis is the result of my own investigations except where otherwise stated. Other sources have been acknowledged by footnotes giving explicit references.

Signed... .....(Candidate)
Date.....15/9/05.....

Statement 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organizations.

Signed.....(Candidate)
Date.....15/9/05.....

Acknowledgements

I would like to thank my supervisor Prof. Oubay Hassan for his time and patience throughout the past three years.

I would like to thank Philip for being there for me throughout the past three years and for helping me through the bad times when there seemed to be no light at the end of the tunnel. (It's a miracle what a hug can do).

Summary

This thesis will investigate surface mesh generation and develop ideas to improve the quality of surface meshes that are currently produced.

Surface geometries are represented by a CAD definition, but the CAD definition does not necessarily guarantee that the surface geometry is acceptable for mesh generation. CAD geometries will often contain a number of detailed features which will need to be improved by processes such as CAD repair before mesh generation can take place. Even then the geometries can still contain problems in the features such as, small sliver surface patches and sliver edges. These features cause major difficulties when meshed, as they generate small distorted elements.

Here we will look to improve the meshes by merging together neighboring surface patches to create a super patch and then generate the mesh on this one surface. The merging of surfaces is controlled by the angle between surface patches.

Another method that will be investigated involves the re-meshing of the geometry based on a prescribed metric.

In addition to looking at this problem of CAD representation we will also look at the growing area of medical imaging. Here we will look to produce a 3D mesh from a set of contours. From this the mesh produced will be re-meshed using the previous ideas to produce a mesh that can be used for analysis.

Contents

1.

Introduction

1.1 Introduction

Before numerical methods the aerodynamic properties of aircraft wings were obtained using wind tunnels. The introduction of numerical methods has enabled the testing of such large components to take place in the confines of the computer. Not that the computer has eliminated the need for full scale testing, as this is still an important part of design. However, what it has enabled is any adjustments to geometries to take place quickly and in turn allows a quick solution to be obtained without the need to totally remake a full-scale model. A key requirement in numerical simulation is mesh generation, where a geometry is discretised into a number of different elements ranging from triangles, quadrilaterals and hexahedral. Recently computational solutions are being obtained for more and more varied situations such as:

- Medical imaging,
- Fluid Flow,
- Structural analysis.

which in turn means that the geometries are becoming more and more complex.

Meshing can be defined as the process of breaking up a physical domain into smaller sub-domains (elements) in order to facilitate the numerical solution of a partial differential equation. While meshing can be used for a wide variety of applications, the principal application of interest is the finite element method and finite volume method. Surface domains may be subdivided into triangle or quadrilateral shapes, while volumes may be

subdivided primarily into tetrahedra or hexahedra shapes. The shape and distribution of the elements is ideally defined by automatic meshing algorithms.

Numerical methods in recent decades have become a mainstay for industrial engineering design and analysis. Increasingly larger and more complex designs are being simulated using numerical methods. With its increasing popularity comes the incentive to improve automatic meshing algorithms.

At the inception of numerical methods, most users were satisfied to simulate vastly simplified forms of their final design utilising only tens or hundreds of elements. Painstaking preprocessing was required to subdivide domains into usable elements. Market forces have now pushed meshing technology to a point where users now expect to mesh complex domains with thousands or even millions of elements with no more interaction than the push of a button.

Before a mesh can be generated, a surface geometry has to be defined. Some of the methods for defining surface geometry are mentioned as follows:-

1.1.1 Geometry Representation

In the late 1950's, hardware known as computer aided manufacturing (CAM) was developed, but there was a lack of adequate software. In order to use CAM, it was necessary to produce a computer-compatible description of the shape being considered. The method considered was that of parametric surfaces.

The theory of parametric surfaces was at the time well understood in differential geometry. Their potential for the representation of surfaces in a computer aided design (CAD) environment were however not known at all. The development of computer aided geometric design (CAGD) started to be explored.

The major breakthrough in CAGD, were undoubtedly the theory of Bezier surfaces and Coons patches, which was later combined with B-spline methods [81]. The development of Bezier curves and surfaces was introduced by P. de Casteljau at Citroen, but his work was never published.

Later P. Bezier at Renault [82] developed the theory of polynomial curves and surfaces. Recently there has been more and more research into surface representation, where NURBS (non-uniform rational B-spline), CAD and STL surface representations have all been developed.

In 1964 J Ferguson [2] introduced cubic splines to the CAGD system. The representation developed by Ferguson is used in this thesis. The curve is expressed in terms of the coordinates of a set of points and their respective tangents. This form is, therefore, particularly suited for curve fitting through a set of points. Other more general forms like, for instance, the Bezier and B-spline representations, define the equation of the curve in terms of a polygon of control points. These forms are preferred for shape design because they permit a better control over the interactive modification of curves. Other types of representation include STL where the surface is defined by a number of triangular elements. The information given includes the element number and node coordinates.

Current procedures for the definition of curved smooth surfaces are mostly based in bivariate parametric representations in which the surface is envisaged as being divided, by a network of parametric lines, into an assembly of topologically rectangular patches. Consequently, each of these patches is bounded by four smooth curves. A variety of different methods can be devised, depending on the procedure employed, to construct the network of parametric lines and the manner in which a surface patch is interpolated from its boundary curves.

The approach employed in this thesis is known as a tensor-product representation of a surface. A topologically rectangular set of points, which are the nodes in the network of parametric lines, is first defined on the surface to be approximated. The parametric lines are, then, obtained by interpolating spline curves through these support points along the two parametric directions. Because the curves are defined using Ferguson representation the surfaces are then defined using Ferguson cubics. The mathematical expression for every surface patch is obtained, in terms of its boundary curves and cross-boundary gradients, by using a generalized Hermitian interpolation between opposite boundaries of the patch. This is known as Coon's technique [5].

1.1.2 Problems of Geometry Definition and CAD Repair

Generating CAD models is classed as the norm, but generally direct analysis on the model is difficult. CAD geometry will often contain a number of detailed features which need to be improved by processes such as CAD repair before mesh generation can take place.

Due to the fact that CAD models are becoming more and more complex there are more problems occurring when reproducing the CAD model for generation. The models used often contain a number of different details for aesthetic and manufacturing purposes only.

Previous to CAD geometry, engineers would build their meshes using bottom-up methods with interactive software that would enable the user to generate arrays of nodes and elements and then stitch them together to form a mesh that closely represented the target geometry. Meshing algorithms now operate on a geometrical model; that is the mesh is generated by some marching process, creating nodes and elements that fill the volumes or faces of the geometrical model. Since the resulting mesh relies highly on the geometry and topology of the original CAD model, a high level of integrity in the geometrical model is required.

Even though geometry top-down mesh generators are classed as the norm, analysts are still looking for ways to improve the CAD geometry for use with their mesh generators. These top-down methods have resulted in the need for updating old CAD models and also getting meshing algorithms to work on the updated and new CAD geometries is difficult. The difficulties arise in the CAD definition, where the geometries often contain a number of discrepancies. These can be forced due to attempts to import entities from another system, or from the CAD definition itself. These discrepancies come in many forms such as:

- Neighbouring edges that are supposed to be joined by a common vertex, but have separate, duplicate points.
- An edge that has end points that do not lie on its path.
- A face whose edges do not form a closed boundary.
- A face that has no finite area.

- Intersecting faces.
- Duplicate edges.
- A domain where faces do not form a closed loop.
- Slivers, in the form of faces and edges.
- Minute edge lengths.
- Narrow faces.
- Triangular shaped surfaces patches, used for aesthetic purposes only.

1.1.3 The Problem

After the geometry has been generated using a CAD model, then CAD repair methods are applied to the geometry to ensure that the model is watertight. Even though CAD repair will eliminate many of the problems highlighted above, there will still remain discrepancies in the model which will cause a poor mesh to be generated in parts of the geometry.

The CAD repair will eliminate many of the topological discrepancies but will not improve the geometry definitions from the mesh generation point of view, hence another method is needed to improve the geometry and eliminate these problems. The methods that will be introduced in Chapters 3 and 4 in this thesis will look to eliminate the following problems.

- Slivers, this is either the surface patches are small or the edges used to represent the geometry are small.
- Narrow surface patches, this is a surface patch where two of the boundary lines come in close contact.
- Badly shaped patches.

These features cause more elements being required to represent the model than is necessary; this can be seen in Figure 1.1 where an aircraft is shown.

1.1.4 Previous Solutions

Previous solutions to the problem involve looking at repairing the geometry via a regeneration process. The models geometric and topological representations need to be improved if the original model representation is not

suitable. Butlin et al [25], Jones et al [24] and Ribo et al [89] look at repairing the geometry via CAD repair methods to eliminate noisy data.

Sheffer [26,27] has looked at introducing an additional layer of new topology on top of the original model to simplify the model. The method is used for simplifications performed in preparation for meshing of geometries. It uses a local analysis approach based on a clustering procedure. This method will eliminate those slivers that are created but will involve the creation of a new topology.

Other work such as that by Marcum [90] looks at using an approximate physical space grid to define the surface during the grid generation process. Then the mapped space coordinates are mapped back to the actual surface at completion.

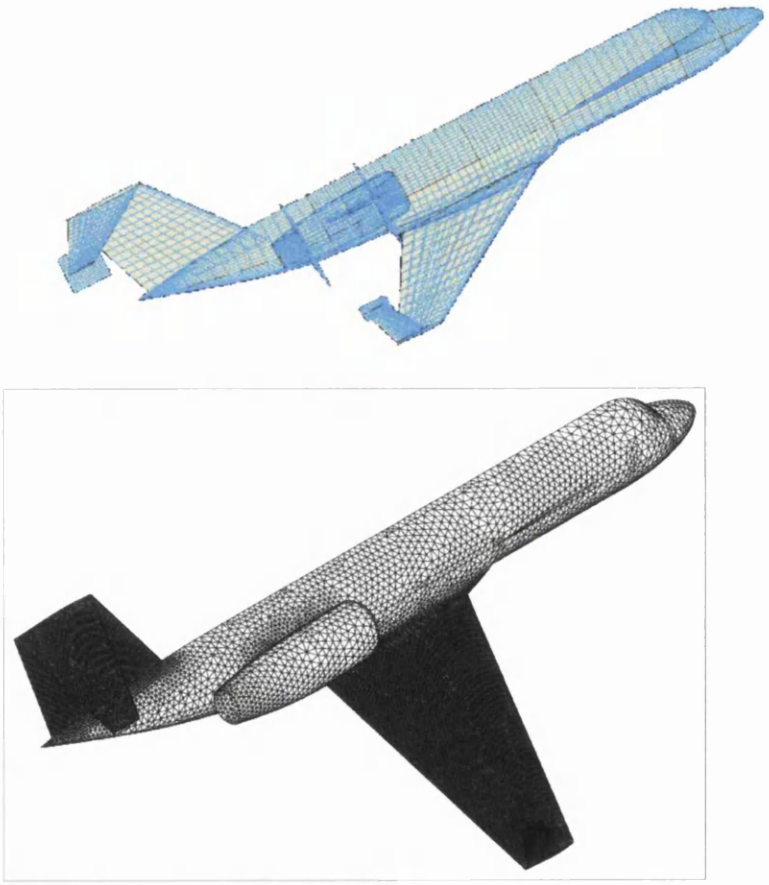


Figure 1.1 Aircraft showing effect of sliver patches.

More recent work by Lee [28] looks at utilising the medial axis to eliminate discrepancies.

1.1.5 Structured or Unstructured Mesh Generation

In structured mesh generation, the domain is divided into a structured assembly of quadrilateral elements. The structure in the mesh can be seen as exactly the same number of mesh elements surrounds each point, see Figure 1.2. The form of a structured mesh allows two directions within the mesh to be identified by a curvilinear coordinate system. Generally, such grids are constructed by mapping the domain of interest into a square and then generating a rectangular mesh over the square. If the equation itself is also mapped, then this grid can be used to obtain a solution, otherwise the inverse mapping is applied to obtain the required mesh over the original domain.

Advantage: Structured mesh generation allows the user to choose an appropriate solution method from among the large number of algorithms which are available. These algorithms are normally implemented in a computationally efficient manner.

Disadvantage: For regions of general shape it is not possible to guarantee an acceptable mesh. However this can be overcome by appropriately sub-dividing the domain into blocks and then producing a grid by applying the mapping method to each block separately. This is better than the previous method but problems can still be caused by the generation of poor quality elements, and by the time necessary to produce a grid for extremely complex domains.

The alternative approach is to divide the domain into an unstructured assembly of elements as shown in Figure 1.3. The main feature of the unstructured mesh is that the number of elements that surround a node are not necessarily constant. Quadrilateral elements could also be used in this method. The unstructured mesh does not lend itself to directionality; therefore, solution techniques based upon this concept (e.g. ADI methods) will not be directly applicable. The methods which are normally used to generate these unstructured meshes are based upon either the Delaunay [31] or the

Advancing Front [9] approaches. Natural candidates for use with unstructured meshes are that of the finite volume and finite element methods.

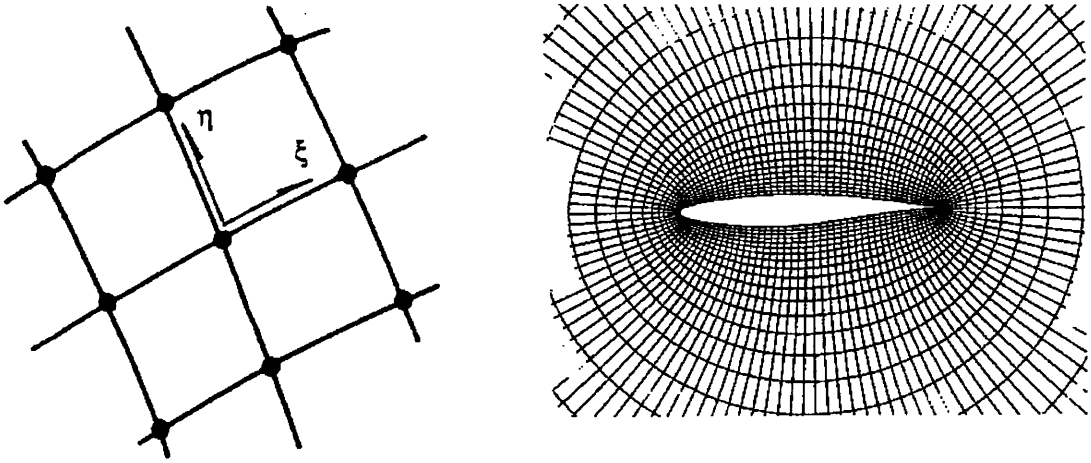


Figure 1.2 Structured mesh.

Advantages: The main advantage of the unstructured mesh generation procedure is that it provides a very powerful tool for discretising domains of complex shape. Also the unstructured mesh naturally offers the possibility of incorporating adaptivity.

Disadvantages: The disadvantages of using the unstructured mesh generation approach are that the number of alternative solution algorithms is currently rather limited, and their computational implementation places large demands on both computer memory and CPU. Also these algorithms are rather sensitive to the quality of the mesh which is being employed and so great care has to be taken during the generation process.

1.1.6 Unstructured mesh generation

The method that will be used throughout this thesis will be that of the unstructured mesh generation process. The reason for this is its strong links with finite element analysis and its ability to handle large complex shapes. Unstructured grids have inherent simplicity of construction in that, by definition, no structure is required. Also it is not inherently necessary to communicate the actual topology of the configuration to the grid generator.

Unstructured mesh generation had its real introduction to the CFD community in the 1980's primarily from Baker [92], Weatherill [105] and Lohner [69].

Due to the growing need for solutions on more complex geometries the need for good quality meshes is more important than ever. The mesh should be sufficiently dense that the numerical approximation is an accurate one, but it cannot be so dense that the solution is impractical to obtain.

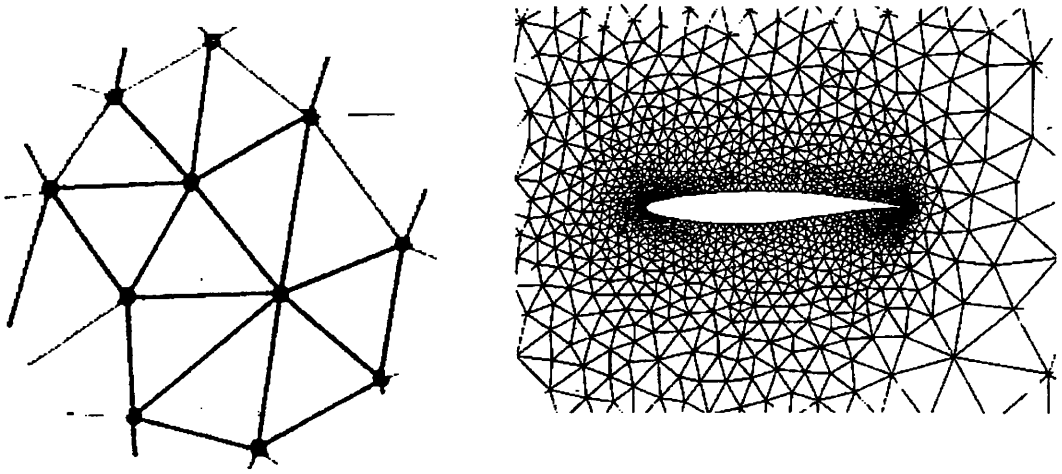


Figure 1.3 Unstructured meshes.

There are a number of different methods that can be used in unstructured mesh generation, these are:

- The Delaunay triangulation
- The Advancing Front method
- Quadtree and Octree meshes
- Hybrid grids
- Hexahedral mesh generation

The methods that we will use in this thesis will be that of the Delaunay triangulation and the Advancing Front method. Here we will look at a brief history of both methods and introduce the main contributors to both methods.

1.1.7 Delaunay Triangulation

Dirichlet [1850] [91], first proposed a method whereby a given domain, in arbitrary space, could be systematically decomposed into a set of packed convex regions [23]. For a given set of points (P_i), the space is subdivided

into regions (V_i), in such a way that the region (V_i) is the space closer to P_i than to any other point. The geometrical construction of tiles is known as the *Dirichlet tessellation.* The tessellation of a closed domain results in a set of non-overlapping convex regions that cover the entire domain. In 1908 Voronoi [106] developed the diagram from the convex regions known as the Voronoi diagram. In 1934 Delaunay [107] developed a method of creating triangles using the Voronoi diagram. Although the Delaunay triangulation was formulated early in the 20th century, it's only recently that the idea has been employed in computer applications. These ideas have developed a wealth of applications ranging from interpolation of data [Farin, 1990] [93], medical imaging [Boissonnat, 1988 [51-52]], computer animation and grid generation [Cavendish, et al., 1985; [94] Shenton and Cendes, 1985; [95] Baker, 1987 [92]; George, et al., 1988; [96,97] Perronnet, 1988; [98] Schroeder and Shephard, 1988; [99] Weatherill and Hassan 1992; [31], Sharov and Nekahashi, 1996, [100]].

1.1.8 Advancing Front Method

The Advancing Front Technique (AFT) for the generation of unstructured triangular meshes was first formulated by George [64], but this original publication did not receive attention. It seems that the first reference to this work appeared in the book written by Thomasset [65] approximately 10 years later. The first journal publication of the method was that of Lo [66], where the AFT was used to produce a triangulation by linking a set of points, which had been generated beforehand in a Cartesian fashion.

The algorithm was modified by Peraire et al. [67], using a new formulation in which elements and points were simultaneously generated. The method also allowed for the generation of high aspect ratio triangles and, more importantly, grid control was introduced through the specification of a spatial variation of the desired element size and shape. This facility was later used for adaptive computations in CFD.

The methodology was subsequently extended to 3D in [68-70]. The use of the AFT for 3D adaptation in compressible flows is described in [71]. Recent implementations of the AFT that improve the generation times and/or the point placement / selection strategies have been reported [72-76]. In

addition, the method has also been modified to produce a procedure for the generation of unstructured meshes of quadrilaterals in [77,78] and of hexahedrals in [79].

1.2 Aims and Objectives

The aim of this thesis is to look at surface mesh generation and develop ideas to improve the quality of surface meshes that are currently produced. As mentioned previously the surface geometries are represented by a CAD definition. But the CAD definition does not guarantee that the surface geometry is acceptable for mesh generation to take place. The CAD geometries will often contain a number of detailed features which will need to be improved by processes such as CAD repair before mesh generation can take place. Even then the geometries can still contain problems in the features such as, small sliver surface patches and sliver edges. These features cause major difficulties when generating meshes, as they generate small sliver elements.

The aim of this thesis is to devise new methods to overcome this problem and produce meshes that do not include these small sliver patches, hence producing meshes of a higher quality. This problem will be looked at in depth and a number of different methods to overcome this problem will be investigated.

Another growing trend in mesh generation is that of medical imaging. As mentioned previously, in 1988 Boisonnat looked at developing this area of research. The process involves taking a scan produced from MRI and producing a 3D surface mesh that can be used for analysis. Therefore as this thesis is involved in surface mesh generation, we will look at this process and link it with the previous method to produce a high quality surface mesh ready for analysis.

1.3 Outline of Thesis

Chapter 2: This chapter introduces the meshing techniques that will be used throughout this thesis, the meshing used will be that of the unstructured type

for 2D, to include the Delaunay triangulation and the Advancing Front Techniques. The chapter begins with the introduction of the Advancing Front method which then goes on to introduce the concepts of the background mesh and the use of sources in mesh generation. The Delaunay triangulation is discussed, which will be used later on in Chapter 5 for the contour mesh generation process, along with mesh quality enhancement techniques which include diagonal swapping and mesh smoothing. Finally, the different types of data handling used throughout this thesis are introduced.

Chapter 3: This chapter begins with looking at the area of mesh improvement and then goes on to look at the history of CAD improvement techniques. In Chapter 2 the meshing process is introduced where we see how the surface geometry is split into a number of individual surface patches that are meshed one at a time using the Advancing Front Technique. This chapter will introduce the idea of the Super Patch process, where neighboring surface patches of similar tangency are merged together to obtain a Super Patch that can be meshed as a whole utilising the Advancing Front Technique. The chapter concludes with a number of examples to show the results of this process compared to the original method.

Chapter 4: This chapter looks at different methods for improving poorly shaped elements that are produced from the small sliver surface patches. The method discussed here is based on a re-meshing algorithm, where the mesh is re-meshed based on a prescribed metric, which can be based on curvature or a specific size chosen by the user. The method involves the use of a G_1 approximation to the surface. The reason for this is to produce a more general method for cases where the surface definition may not be available. Mesh quality techniques are used to validate the results obtained in both Chapters 3 and 4. The chapter shows a number of examples produced showing the elimination of the poorly shaped elements as related to Chapter 3. The chapter then goes on to show some further applications of the method, including medical images and size specified meshes.

Chapter 5: This chapter will introduce the concept of producing a 3D surface mesh from a set of MRI scans produced for a number of medical images, although this process can be used for any scanned image. First a number of previous solutions are introduced and then the method to be used here is shown. Where the Delaunay triangulation is used to produce a mesh, the boundary nodes for each individual scan are joined via a closest node procedure to produce the 3D surface mesh. The method utilizes the Delaunay and Voronoi relationship to produce the examples shown at the end of the chapter.

Chapter 6: This chapter presents the conclusions resulting from the thesis and recommends possible areas of further research.

2.

Unstructured Mesh Generation Techniques

2.1 Introduction

In this chapter a review of the different methods of unstructured mesh generation used throughout the thesis will be presented. In an unstructured mesh, the number of points and elements which are neighbours to an interior point, is not kept constant throughout the domain. The lack of regularity in the mesh representation offers great versatility and geometrical flexibility to the mesh generating process. The mesh generation methods used throughout have a number of characteristics, such as an ability to handle arbitrary geometries in a fully automatic manner with minimum user intervention. The scheme provides control over the spatial variation of element size and shape through the domain and also allows adaptive methods to be used to produce the most accurate approximation of the solution for any given number of points.

Two main methods of unstructured mesh generation methods that are used in this thesis are that of the Advancing Front method and the Delaunay triangulation. This chapter will describe both methods in detail.

2.2 Unstructured Surface Mesh Generation by the Advancing Front Method

One of the distinctive features of this method is that elements and points are generated at the same time. The boundary of the domain to be meshed is analytically represented by means of parametric composite curves

with surfaces [10] fitted, ensuring second order continuity through an appropriate set of data points on the boundary. The representation provides a mathematical definition that makes the handling of complex geometrical domains possible.

The characteristics of the mesh, such as the element size and shape in the region which is to be discretized, are defined in terms of parameters which are functions of the position. This allows elements to be generated with varying size with the possibility to be stretched, if required, along certain directions.

The control of the mesh characteristics is obtained by the use of a background mesh, which specifies a suitable spatial distribution of the mesh parameters. This is done by defining a mesh of triangles in 2D or tetrahedra in 3D. With the values of the mesh parameters specified at the nodes, a linear variation inside the element is assumed. The use of the background mesh gives a larger control of the mesh to be generated and hence facilitates the construction of adaptively regenerated meshes in steady-state problems. In this section the Advancing Front method will be described together with the curve and surface definitions.

2.2.1 Characterisation of the Mesh: Mesh Parameters

The geometrical characteristics of the mesh are locally defined by means of the mesh parameters, which control the size and shape of the generated elements. If N denotes the number of dimensions, the mesh parameters are chosen to be a set of mutually orthogonal directions $\alpha_i; i = 1,...N$, and N associated element sizes $\delta_i; i = 1,....N$. Therefore if at a point all the element sizes are equal then the mesh within the region of the point will contain equilateral elements. The mapping is linear and transforms the space where the element is generated (stretched space) into another space (normalised space) where the elements will be approximately equilateral with unit average size. The transformation (T) that takes place is represented by an N × N matrix, which for a general mesh is a function of position.

$$T(\alpha_i, \delta_i) = \sum_{i=1}^{N} \frac{1}{\delta_i} \alpha_i \otimes \alpha_i$$

The mechanism employed is to define the required spatial distribution of the mesh parameters. This is achieved by the use of a background mesh and/or by the use of sources.

2.2.2 Mesh Control: Background Mesh

The background mesh is made up of triangles in 2D and tetrahedra in 3D. The mesh must completely cover the region that is to be discretized, Figure 2.1. The background mesh consists of a small number of elements, although a single element can be used to impose linear or constant spacing and stretching through the computational domain. Nodes are located at the element vertices and values for α_i and δ_i are defined at these nodes. The generation process takes place in the normalised space where the elements are unity. The transformation (T) is used to map the points in the physical space into the normalised space where the element can be generated and the coordinates for the new point to be created. Then using the inverse of the transformation the points are transferred back into the physical space.

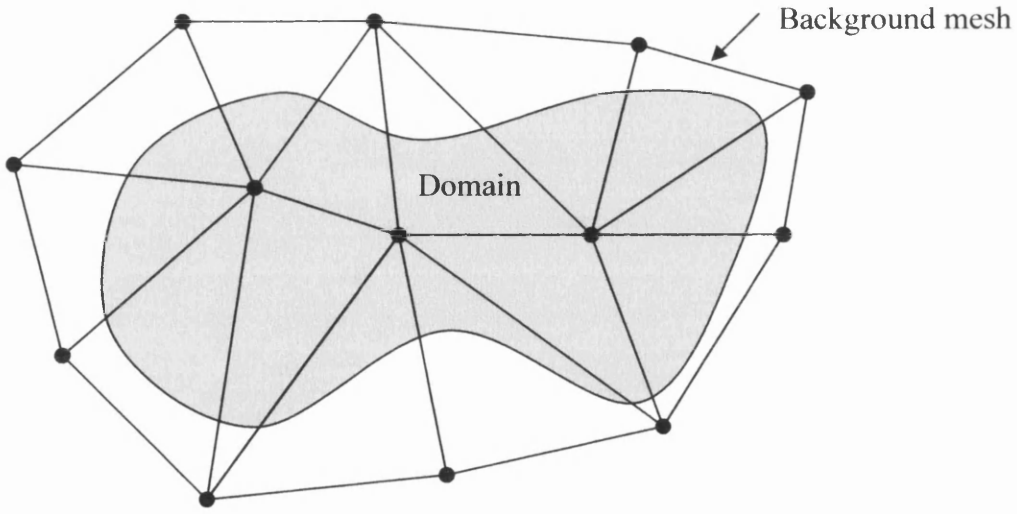


Figure 2.1 The background mesh for the specification of a spatial distribution of mesh parameters.

2.2.3 Mesh Control: Sources

The requirement of constructing an adequate background grid, for complex two-dimensional geometries and simple 3D geometries, has proved to be a significant barrier to the successful use of the approach by the inexperienced user [9]. The concept of point, line and triangle sources is used to define a specified size variation at any point in the domain. With the location of the source specified, the nodal spacing δ defined by the source at a location x is determined as

$$\delta(x) = \delta 1 \qquad |x1| \lhd R1$$
$$\delta(x) = \delta 1 \exp(|x1| \ln[2/R2 - R1]) \qquad |x1| \lhd R1$$

where $|x1|$ denotes the distance from x to the point, line, triangle source and $\delta 1$, *R1* and *R2* are user specified constants. Throughout the process the spacing at any point x is taken as the minimum defined at the location by the background grid and by all the active sources. During the simulation of inviscid aerodynamic flow, it is known that suitable computational grids need to be clustered in certain regions of the domain. Clustered regions are used near the leading and trailing edges of wings, while larger elements can be used else where in the mesh. Meshes of this type can be easily generated to conform to any source; an example of the sources used can be seen in Figure 2.2. An example of a mesh after the application of sources is shown in Figure 2.3 where a wing of an aircraft with line sources placed on the leading and trailing edge have been used to generate the final mesh.

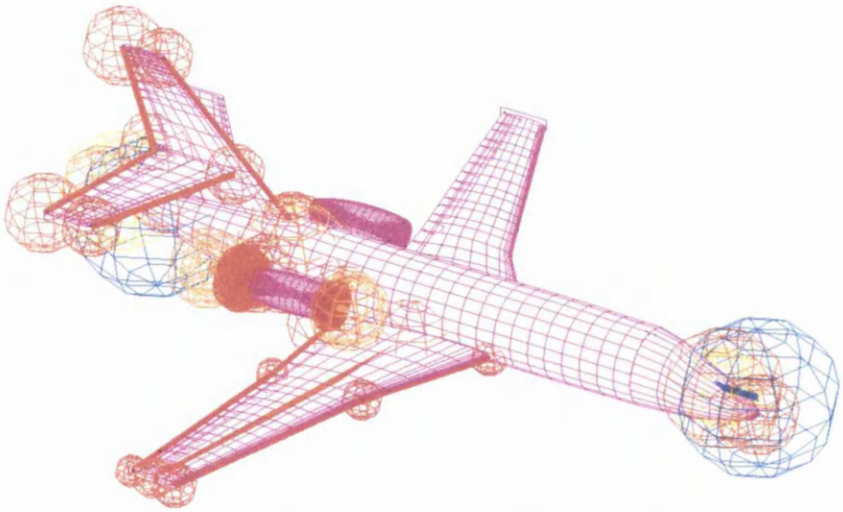


Figure 2.2 Sources applied to aircraft.

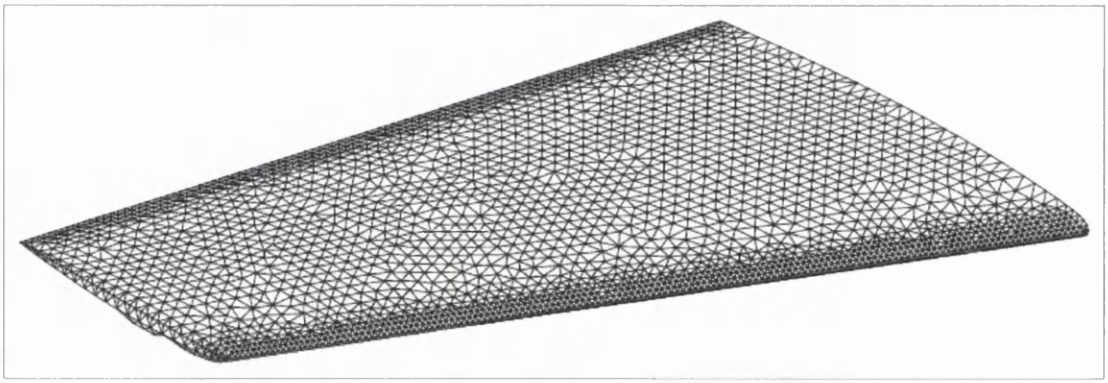


Figure 2.3 Example of line source on an m6 wing.

2.2.4 Domain Representation

A precise mathematical definition of the geometry of the domain to be discretized is an indispensable requirement for any mesh generation technique. If automatic discretisation of an arbitrary domain is to be achieved, the mathematical descriptions of their topology should possess the greatest possible generality. This provides a sound computer implementation in which any geometrical quantity relevant to the generation procedure can be automatically computed. Solid modelling provides the most general up-to-date set of methods for the computational representation and analysis of general shapes matching the above requirements.

Following the ideas of solid modelling, the domain to be considered has a 3D region bounded by faces that are curved surfaces and edges which are curves. In the following, these items will be referred to as *boundary faces* and *boundary edges*, respectively. This is shown in Figure 2.4, where a three dimensional domain is represented by its geometrical components. The approximate representation of the boundary components is accomplished by means of composite curves and surfaces.

2.2.5 Curve Representation

A cubic composite curve can be formulated in various different ways such as cubic splines, NURBS or Ferguson curves [1]. The applications that are used in the thesis involve the Ferguson representation. In the Ferguson representation [2], the curve is expressed in terms of coordinates of a set of points and their respective tangents. The Ferguson representation is

therefore, particularly suited for curve fitting through a set of points. Other more general forms such as the Bezier and B-spline [4] representations define the equation of the curve in terms of a polygon of control points. These forms are preferred for design as they allow a better control over the interactive modification of curves. Within this thesis the interest lies in the representation of a domain (e.g. an aircraft) whose geometry is given. Therefore, as they all produce the same interpolation curve when fitted through a set of points, the Ferguson representation, which is the simplest, is used for the surface generator.

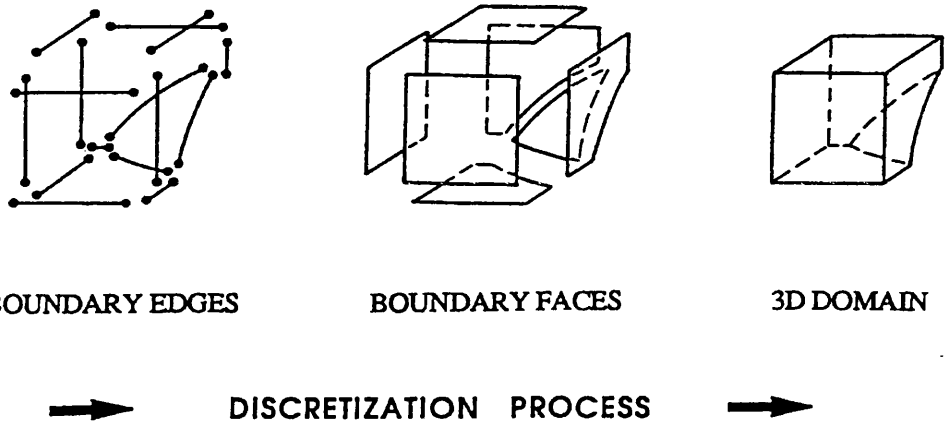


Figure 2.4 Decomposition of the boundary of a three-dimensional domain into its surface and curve components.

2.2.6 Ferguson Curve Representation

The parametric definition of a curve consists of a piecewise interpolation of cubic polynomials through an ordered set of data points. The order in which these points are given defines the orientation as described previously. In the Ferguson representation, each cubic polynomial is expressed, in terms of the position and the tangent vectors at the two end points, as

$$\underline{r}(v) = (1, v, v^2, v^3) C \begin{bmatrix} \underline{r}^{(1)} \\ \underline{r}^{(2)} \\ \underline{t}^{(1)} \\ \underline{t}^{(2)} \end{bmatrix} \qquad 0 \le v \le 1 \qquad (2.5.1)$$

where $\underline{r}^{(1)}$ and $\underline{r}^{(2)}$ are the coordinates of the end points of the segment, $\underline{t}^{(1)}$ and $\underline{t}^{(2)}$ are their respective tangents, and *C* is a constant matrix given by

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & 1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \tag{2.5.2}$$

The tangent to the curve is computed as the differential of the coordinates,

$$\underline{t}(v) = \underline{r}'(v) = \frac{d\underline{r}(v)}{dv} \tag{2.5.3}$$

The number and distribution of the data points for each segment should be given in such a manner that the interpolated curve accurately approximates the intersection of the corresponding surface components. The interpolation problem, which is illustrated in Figure 2.6, consists of fitting a parametric spline, defined in a piecewise manner, through a set of *n* points $\underline{r}_j; j = 1,....n$. At interior points, continuity of slopes is guaranteed for any choice of the tangent vectors, provided that a unique tangent vector is used for the definition of the two adjacent cubic segments. The unknowns of the problem are the tangents at the points $\underline{t}_j; j = 1,...n$. Therefore by imposing continuity of curvature at the interior nodes, one obtains a set of *n* -2 equations of the form.

$$t_{j-1} + 4t_j + t_{j+1} = 3(r_{j+1} - r_{j-1}) \qquad j = 2,....., n-1 \tag{2.5.4}$$

The two additional relations required to solve the problem are obtained by specifying the boundary conditions at the end points to consist of zero curvature.

2.2.7 Ferguson Surface Representation

Current procedures for the definition of curved smooth surfaces are mostly based in bivariate parametric representations in which the surface is envisaged as being divided, by a network of parametric lines, into an assembly of topologically rectangular patches as shown in Figure 2.7, where a rectangular set of data points $\underline{r}jk$; are used to define the surface with *j=1,...m; k=1,...n.* Consequently, each of these patches is bounded by four smooth curves. A variety of different methods can be devised, depending on

the procedure employed to construct the network of parametric lines and the manner in which a surface patch is interpolated from its boundary curves.

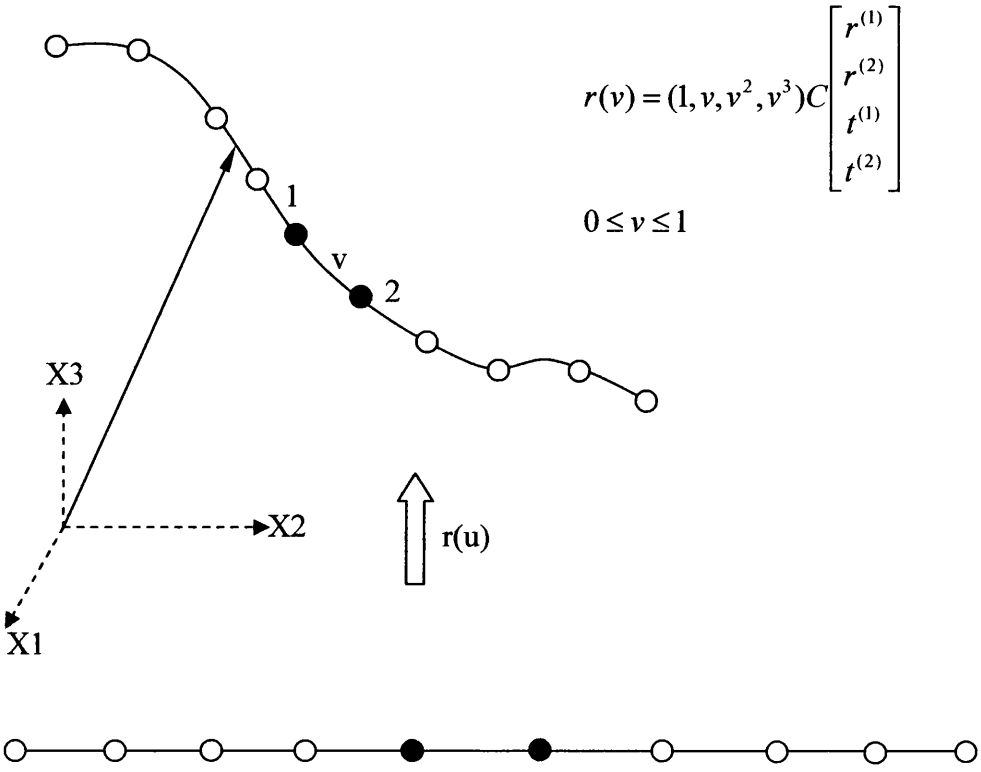


Figure 2.6 Interpolation of a piecewise cubic spline through a set of data points.

The approach used above is known as a tensor-product representation of a surface. Two families of parametric lines are obtained by interpolating spline curves, first through the points with constant *j* and then through the points of constant *k*. The procedure used to interpolate each boundary curve was described in the previous section. For every quadrilateral surface patch its mathematical expression is given in terms of the four cubic curves that form its boundary and the twist vector at the four corner points.

$$\underline{r}(v,w) = (1, v, v^2, v^3)C \begin{bmatrix} \underline{r}^{(1)} & \underline{r}^{(4)} & \underline{r}_{,w}^{(1)} & \underline{r}_{,w}^{(4)} \\ \underline{r}^{(2)} & \underline{r}^{(3)} & \underline{r}_{,w}^{(2)} & \underline{r}_{,w}^{(3)} \\ \underline{r}_{,v}^{(1)} & \underline{r}_{,v}^{(4)} & \underline{r}_{,vw}^{(1)} & \underline{r}_{,vw}^{(4)} \\ \underline{r}_{,v}^{(2)} & \underline{r}_{,v}^{(3)} & \underline{r}_{,vw}^{(2)} & \underline{r}_{,vw}^{(3)} \end{bmatrix} C^T \begin{bmatrix} 1 \\ w \\ w^2 \\ w^3 \end{bmatrix} \qquad (2.5.5)$$

Where C is the matrix as described in equation (2.5.2), C^T is the transpose, and the comma denotes partial differentiation, where,

$$\underline{r}_{,v} = \frac{\partial r}{\partial v}; \qquad \underline{r}_{,w} = \frac{\partial r}{\partial w}; \qquad \underline{r}_{,vw} = \frac{\partial^2 r}{\partial v \partial w}. \tag{2.5.6}$$

The notation employed to denote the corner points of the patch are:

$$\underline{r}^{(1)} = \underline{r}(0,0); \quad \underline{r}^{(2)} = \underline{r}(1,0); \quad \underline{r}^{(3)} = \underline{r}(1,1); \quad \underline{r}^{(4)} = \underline{r}(0,1). \tag{2.5.7}$$

This representation uses a Hermite interpolation between the opposite boundaries of the patch [2]. The twist vectors $(\underline{r}_{,vw})_{jk}$ at the corner points are computed so that the overall second order continuity is achieved on the interpolated surface.

For convenience, global parametric coordinates u^1 and u^2 are defined. For the patch *(j,k)* these coordinates are related to the local patch coordinates *v* and *w* according to $u^1 = v + j - 1$ and $u^2 = w + k - 1$. In this way, a global mapping $\underline{R}(u^1, u^2)$ is established between the rectangular region in the parametric plane defined by $0 \le u^1 \le p$; $0 \le u^2 \le q$ and the tensor product surface. The orientation of the surface is defined by specifying the outward normal which points towards the region to be discretized.

2.2.8 Curve Discretisation.

The discretisation of the boundary curve components is achieved by positioning nodes along the curve according to a spacing dictated by the local value of the mesh parameters. Consecutive points are joined by straight lines to form sides. In order to determine the position and number of nodes to be created on each curve component, a number of steps are followed including:

- Subdividing each cubic segment into smaller cubic segments until their length is smaller than a certain prescribed value. This is normally taken as the minimum spacing specified in the background mesh. The length of each segment is computed numerically. When subdividing a cubic segment, the position and tangent vectors corresponding to every new point can be found directly from the original definition of the segment.

- For every data point $\hat{\underline{r}}_j; j = 1,....,n$, that is the points used to define the curve and those created to satisfy the maximum length criterion, interpolate from the background mesh the coefficients of the transformation T_j and transform the position and tangent vectors i.e. $\hat{\underline{r}}_j = T_j \underline{r}_j$ and $\hat{\underline{t}}_j = T_j \underline{t}_j$. The new position and tangent vectors $\hat{\underline{r}}_j, \hat{\underline{t}}_j; j = 1,...,n,$ define a spline curve which can be interpreted as the image of the original curve component in the normalised space.

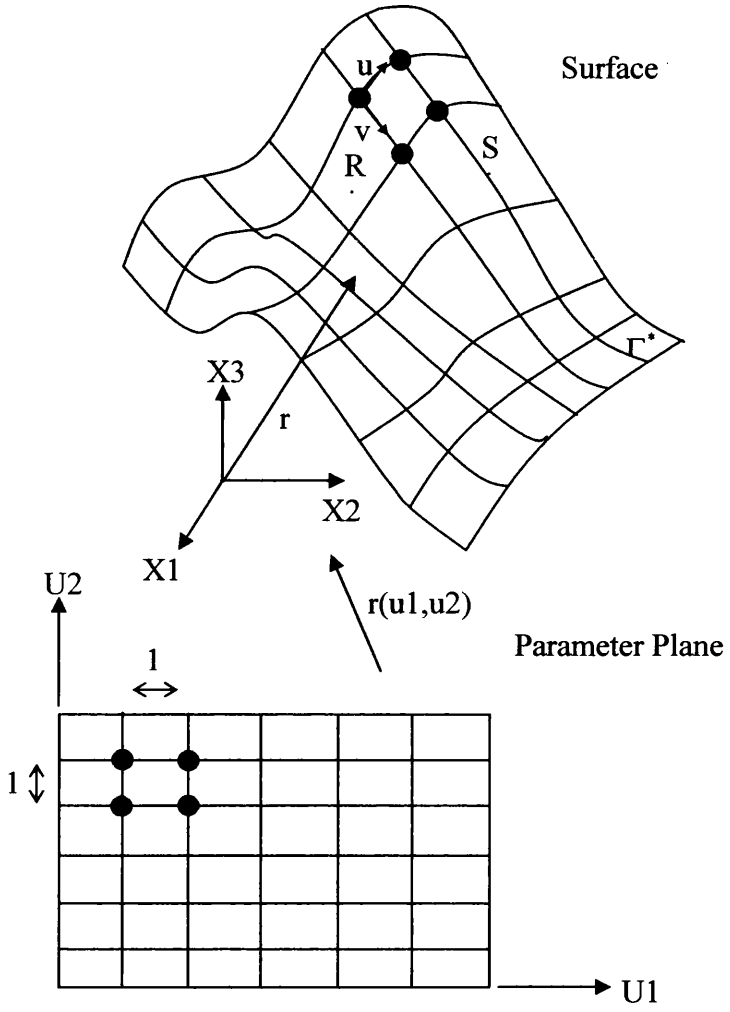


Figure 2.7 Interpolation of a composite surface through a set of data points.

- Compute the length of the curve in the normalised space and subdivide it into segments of approximately unit length. For each newly created point, calculate the cubic segment in which it is contained and its parametric coordinate. This information is used to determine the

coordinates of the new nodes in the physical space, using the curve component definition.

2.2.9 Surface Discretisation.

The method followed for the triangulation of the surface components is an extension of the mesh generation procedure for planar domains described below. Each individual surface patch is discretized by generating a two dimensional mesh of triangles in the parametric plane (u^1,u^2) and then these triangles are mapped into the physical space by the mapping process defined in section 2.2.7. This mapping process produces a one to one correspondence between the surface components and a region on the parametric plane (u^1,u^2), Figure 2.8. The mesh produced in the parametric plane will be transformed, via the mapping $\underline{r}(u^1,u^2)$ as described in section 2.2.7, into a valid triangulation into the physical space. The construction of the triangular mesh in the parametric plane using the two dimensional mesh generator requires the determination of an appropriate spatial distribution of the two dimensional mesh parameters. These consist of a set of two mutually orthogonal directions $\underline{\alpha}_i^*; i=1,2$, and two associated element sizes $\underline{\delta}_i^*; i=1,2$.

Following the example in Figure 2.9, consider a point P in the parametric plane of coordinates (u^1,u^2) where the values of the mesh parameters $\delta_i, \alpha_i; i=1,2$ are to be computed. Its image on the surface will be the point P^* of coordinates $\underline{r}(u^1,u^2)$. The values of the three dimensional mesh parameters δ_i^*, α_i^*; $i=1,2,3$ at this point are obtained by means of interpolation from the background mesh. At the present stage, it is possible to calculate the required values of the 2D mesh parameters by considering the stretching directions and their respective spacings in the tangent plane to the surface. However, in order to simplify the procedure, it is more convenient to apply the auxiliary stretched-unstretched mapping T. After this transformation the spacing at point P**, the image of P*, is constant and equal to one. The

transformation T_p between the physical space and the normalised space at the point P can be defined as:

$$\underline{R}(u^1,u^2) = T_p \underline{r}(u^1,u^2) \tag{2.5.8}$$

A curve in the parametric plane passing through the point P and with unit tangent vector $\underline{\beta} = (\beta^1,\beta^2)$ at this point, is transformed by the above mapping into a curve in the normalised space passing through the point $T_p(P)$. The arc length parameters ds and dz, along the original and transformed curves respectively, are related by the expression

$$(d\zeta)^2 = \left\{ \sum_{i;j=1}^{2} \frac{\partial \underline{R}}{\partial u^i} \frac{\partial \underline{R}}{\partial u^i} \beta^i \beta^j \right\} (ds)^2 \tag{2.5.9}$$

Assuming that this relation between the arc length parameters also holds for the spacings, the spacing δ_β can be computed along the direction $\underline{\beta}$ in the parameter plane as

$$\delta_\beta = \sqrt{\sum_{i;j=1}^{2} \sum_{i;j=1}^{2} \frac{\partial \underline{R}}{\partial u^i} \frac{\partial \underline{R}}{\partial u^i} \beta^i \beta^j} \tag{2.5.10}$$

The two dimensional mesh parameters $\underline{\alpha}^i, \delta^i, i = 1,2$ are determined from the directions in which δ_β attains an extremum. This reduces to finding the eigenvalues and eigenvectors of a symmetric 2x2 matrix.

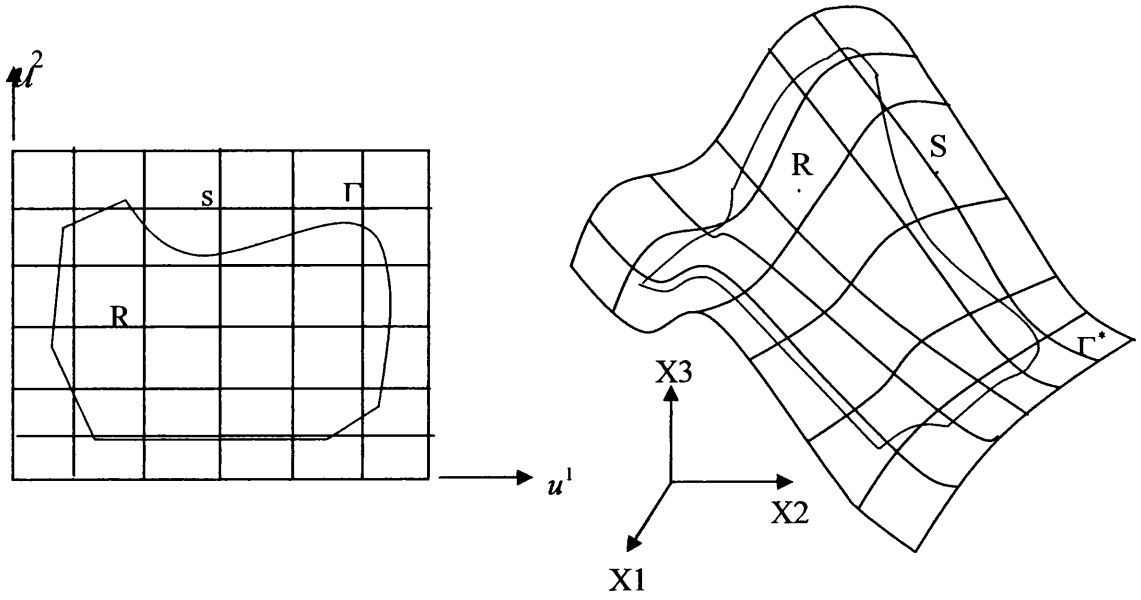


Figure 2.8 Mapping of a surface component onto a two dimensional domain.

To form the initial front, the (u^1,u^2) coordinates of the nodes already generated on the boundary curve components have to be computed. As the mapping $\underline{r}(u^1,u^2)$ cannot be inverted analytically, the coordinates (u^1,u^2) of such points are found numerically by using a direct iteration procedure.

2.3 The Advancing Front

The Advancing Front Technique AFT, for the generation of unstructured triangular meshes was first formulated by George [64], but this original publication did not receive significant immediate attention. It seems that the first reference to this work appeared in an appendix of the book by Thomasset [65]. The first journal publication of the method was that of Lo [66], where the AFT was used to produce a triangulation by linking a set of points, which had been generated beforehand in a Cartesian fashion. The algorithm was modified by Peraire et al. [67], using a new formulation in which elements and points were simultaneously generated. The method also allowed for the generation of high aspect ratio triangles and, more importantly, grid control was introduced through the specification of a spatial variation of the desired element size and shape. This facility was later used for adaptive computations in computational fluid dynamics.

2.3.1 The Advancing Front Technique

The generation problem consists of subdividing an arbitrarily complex domain into a consistent assembly of elements. The consistency of the generated mesh is guaranteed if the generated elements cover the entire domain and the intersection between elements occur only on common points, sides or triangular faces in the three dimensional case. The final mesh is constructed in what can be called a bottom-up-manner. This means that the process starts by discretising each boundary curve in turn. Nodes are then placed on the boundary curves and then contiguous nodes are joined by straight line segments. These sides become sides of triangular faces. The length of these segments have to be consistent with the desired local

distribution of mesh size. This operation is repeated for each boundary curve in turn.

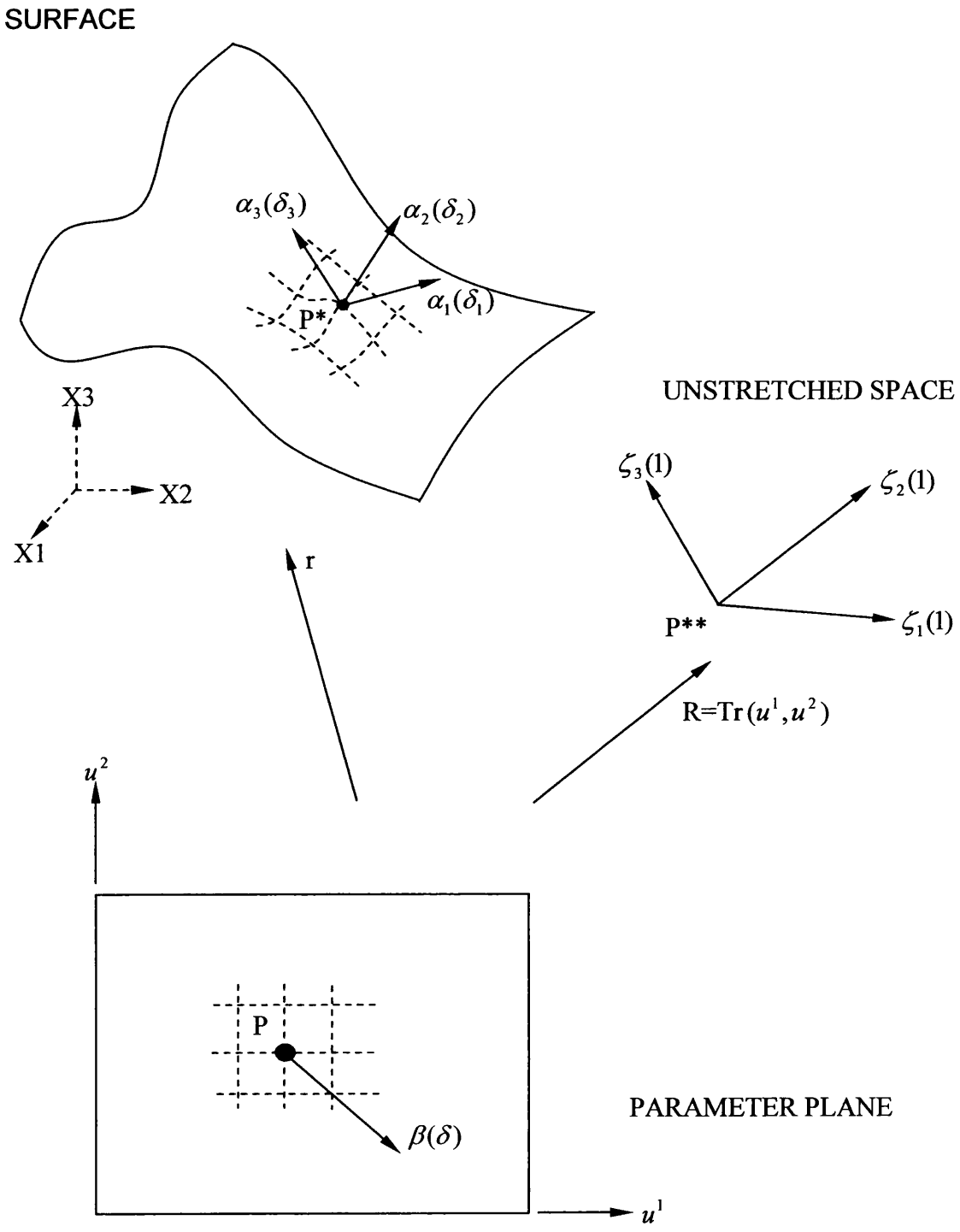


Figure 2.9 Computation of the mesh characteristics in the parameter plane (u^1,u^2).

The next stage consists of generating planar faces. For each two-dimensional region or surface to be discretized, all the sides produced when discretising its boundary curves are assembled into the so-called initial front. The relative orientation of the curve components with respect to the surface must be taken into account in order to give the correct orientation to the sides in the initial front. This front is used to generate a triangular mesh on the surface. The size and shape of the generated triangles must be consistent with the local desired size and shape.

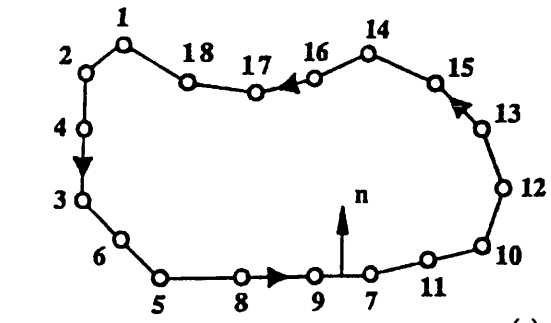
2.3.2 Front Updating

The triangle generation algorithm utilizes the concept of a generation front. The front is a dynamic data structure that changes continuously during the generation process. At the start of the process the front consists of the sequence of straight line segments that connect consecutive boundary nodes. At any given time, the front contains the set of all the sides which are currently available to form a triangular face. Any straight line segment that is available to form an element side is termed active, whereas any segment no longer active is removed from the front. During the generation process an active side is selected from the front and a triangular element is generated. This may involve creating a new node or simply connecting to an existing one. After the triangle has been generated, the front is updated. This updating process is illustrated in Figure 2.10.

Thus while the domain boundary will remain unchanged, the generation front changes continuously and needs to be updated whenever a new element is formed. The generation proceeds until the front is empty. Figure 2.11, illustrates the idea of the Advancing Front Technique for a circular planar domain by showing the initial front and the form of the mesh at various stages during the generation process.

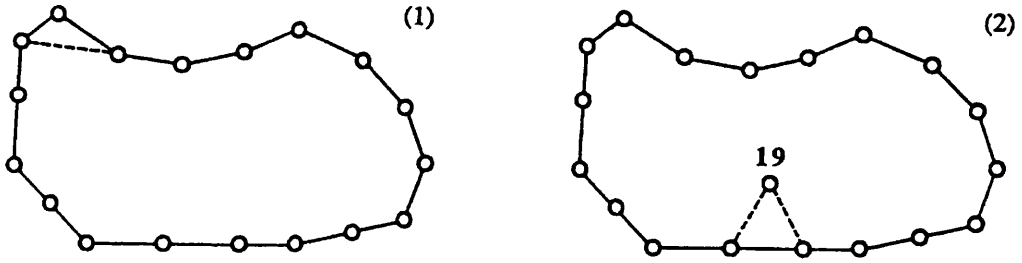
2.3.3 Triangle Generation in 2D

In the process of generating a new triangle the following steps are involved (Figure 2.12).

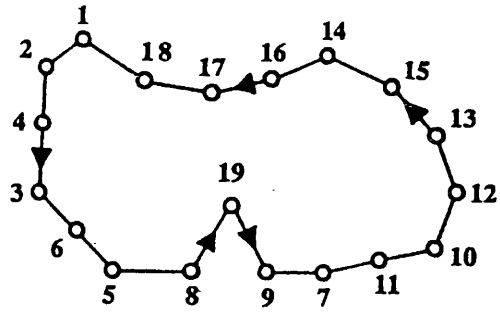


1	3	2	4	6	5	8	9	7
2	6	4	3	5	8	9	7	1
11	12	10	13	15	16	14	17	1
10	13	12	15	14	17	16	18	1

(a)



(b)



ELIMINATED

1	3	2	4	6	5		9	7	AI
2	6	4	3	5	8		7	11	
11	12	10	13	15	16	14	17	18	8
10	13	12	15	14	17	16	18	1	19

(c)

Figure 2.10 The front updating procedure in two dimensions.

(a) The initial generation front.

(b) Creation of a new element. (1) no new point is created; (2) new point 19 is created.

(c) The updating of the front for the case(b)(2).

Algorithm

1. Select a side AB in the front, the method used is to choose the shortest side in the front, to be used as a base for the triangle to be generated.
2. Interpolate from the background mesh the transformation matrix **T** at the centre of the side M and apply it to the nodes in the front that are relevant to the triangulation. In this implementation the relevant points can be defined by all those that lie inside the circle of centre M and radius three times the length of the side being considered. Let $\overline{A}$, $\overline{B}$ and $\overline{M}$ denote the positions in the normalized space of the points *A*, *B*, and *M*, respectively.

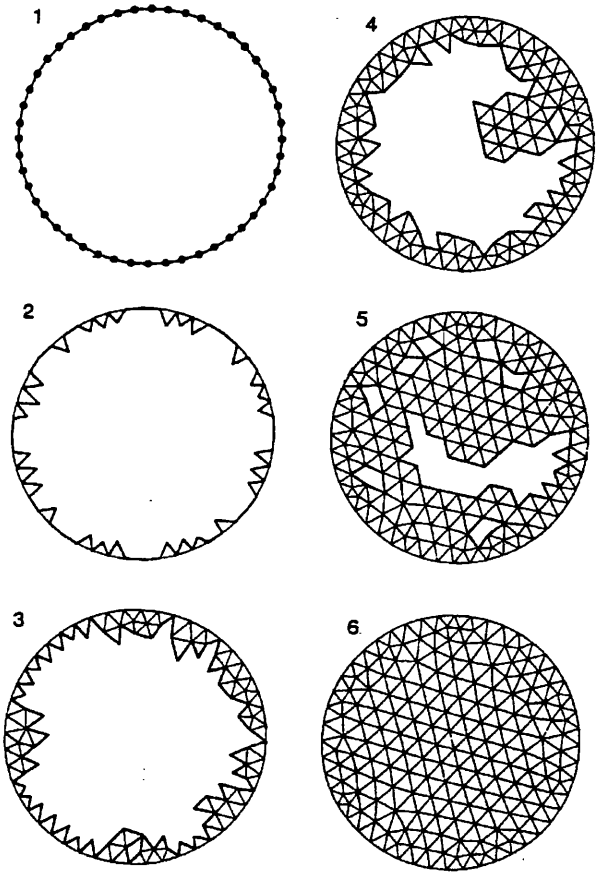


Figure 2.11 The Advancing Front Technique showing different stages during the triangulation.

3. Determine, in the normalized space, the ideal position $\overline{P}_1$ for the vertex of the triangular element. The point $\overline{P}_1$ is located on the line perpendicular to the side that passes through the point $\overline{M}$ and at a distance δ_1 from the points $\overline{A}$ and $\overline{B}$. The direction in which $\overline{P}_1$ is generated is determined by the orientation of the side. The value δ_1 is chosen according to the criterion, where L is the distance between points $\overline{A}$ and $\overline{B}$.

$$\delta_1 = \begin{cases} 1.00 & if\ 0.55 \times L < 1.00 < 2.00 \times L \\ 0.55 \times L & if\ 0.55 \times L < 1.00 \\ 2.00 \times L & if\ 1.00 > 2.00 \times L \end{cases} \qquad (2.5.11)$$

Only in situations where the side AB happens to have characteristics very different from those specified by the background mesh, will the value of δ_1 be different from unity. However, the above inequalities must be taken into account to ensure geometrical compatibility.

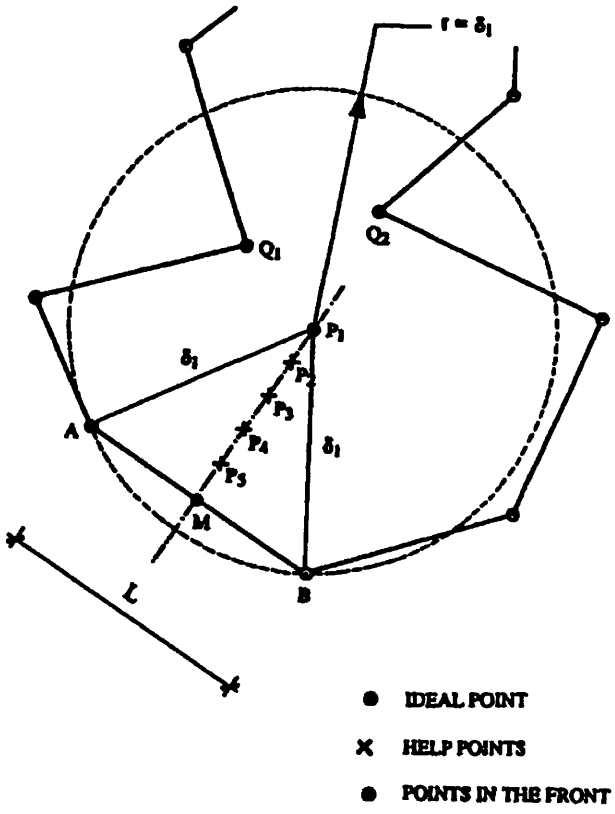


Figure 2.12 Generation of new triangle.

4. Select other possible candidates for the vertex and order them in a list. Two types of points are considered viz. (a) all nodes $Q_1, Q_2, \ldots$ in the current generation front that are, in the normalized space, interior to a circle with center $\overline{P_1}$ and radius $r = \delta_1$, and (b) the set of points $P^1, \ldots, P^5$ generated along the height P^1M. For each point Q_i, construct the circle, with center C_Q^i on the line defined by points P^1 and M, and that passes through the points Q_i, *A* and B^1. The position of the centers, C_Q^i, of these circles, on the line P^1M, defines an ordering of the Q_i points. A list is created that contains all the Q_i points in which the point with the furthest center from P^1 in the direction P^1M appears at the head of the list. The points $P^1, \ldots, P^5$ are added at the end of this list.
5. Select the best connecting point. This is the first point in the order list which gives a consistent triangle. Consistency is guaranteed by ensuring that none of the newly created sides intersects with any of the existing sides in the front.
6. Finally, its coordinates in the physical space are obtained by using the inverse transformation T^{-1}.
7. Store the new triangle and update the front by adding/removing the relevant sides.

2.4 Delaunay Triangulation

In 1850 Dirichlet proposed a method whereby a given domain could be systematically decomposed into a set of packed convex polygons. Where two points in the same plane, P_i and P_j, could be split into two regions, V_i and V_j by the perpendicular bisector of the line joining the two points. The region V_i is the space closer to P_i than to P_j and vice versa. These ideas can be extended further, by stating that for any given set of points in the plane, the

regions V_i are territories which can be assigned to each point such that V_i represents the space closer to P_i than to any other point in the set. This construction of tiles is known as the Dirichlet tessellation. This tessellation of a closed domain results in a set of non-overlapping convex polygons called Voronoi regions, which cover the whole domain. A Voronoi region $\{V_i\}$ represents the set of all points that are closer to P_i than to any other point. The region $\{V_i\}$ is a Voronoi polygon.

For two dimensions it can be seen that the territorial boundary which forms a side of a Voronoi polygon must be midway between the two points which it separates and is therefore the perpendicular bisector of the line joining the two points. If all point pairs which have the some segment of boundary in common are joined by straight lines, then the result is a triangulation of the convex hull of all points in the domain. This triangulation is known as the Delaunay triangulation. An example of a two dimensional Delaunay triangulation is shown in Figure 2.13.

The Delaunay triangulation has some rather interesting properties:

1. One of particular interest is the so-called in-circle criterion, where the vertices of the Voronoi diagram are at the circumcentres of the circles which pass through the three points which form each triangle.
2. From the Dirichlet definition it follows that no points, other than the so-called forming points which from the triangles, fall within the circles.

These can be seen in Figure 2.14.

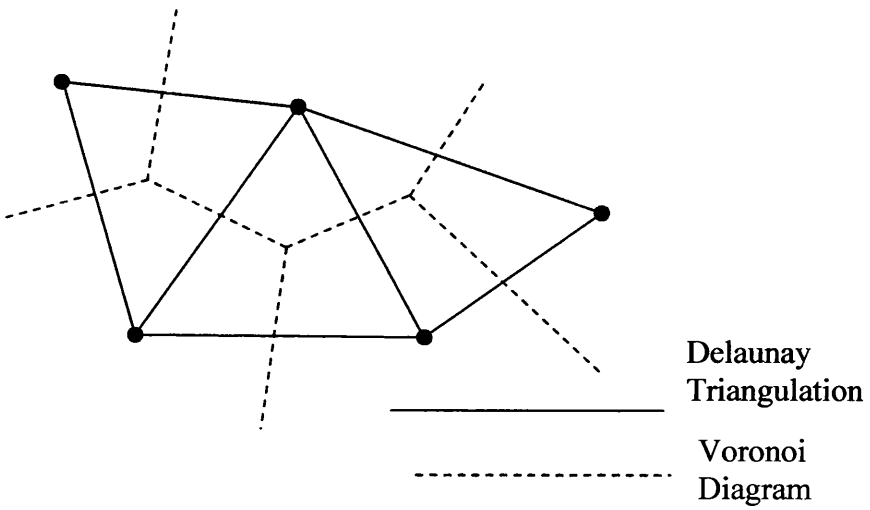


Figure 2.13 The Delaunay and Voronoi constructions.

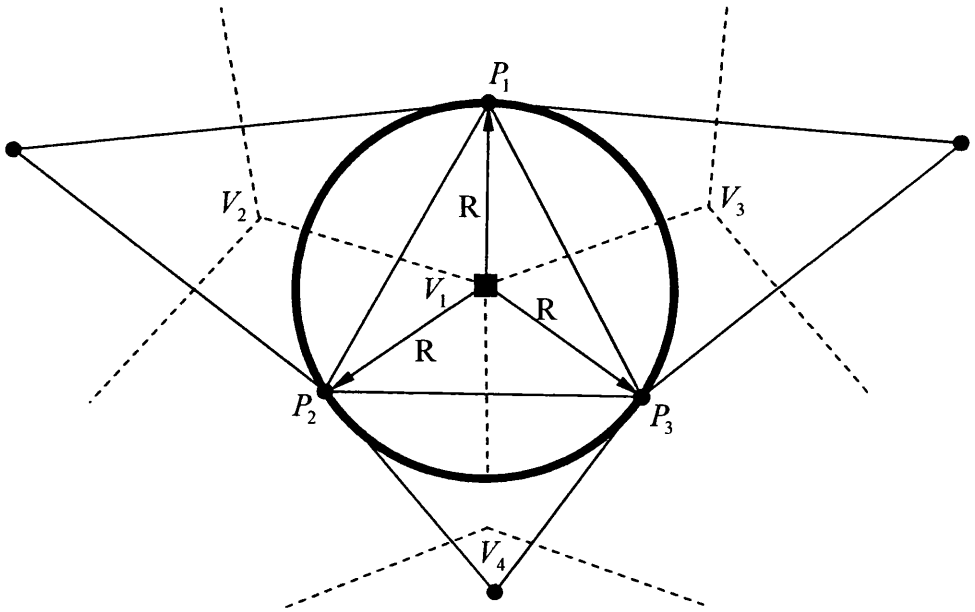


Figure 2.14 The Delaunay and Voronoi constructions.

Algorithm to Construct the Delaunay Triangulation

The in-circle criterion forms the basis for the most popular algorithm for the construction of the tessellation which was proposed by Bowyer and

Watson [29, 30]. The algorithm is applicable in both two and three dimensions.

The basic outlines of the algorithm, presented here for two dimensions are;-

1. Define the convex hull within which all points will lie. It is appropriate to specify 4 points together with the associated Voronoi diagram.
2. Introduce a new point anywhere within the convex hull $\{1,.....,4\}$.
3. Determine all vertices of the Voronoi diagram to be deleted. A point which lies within the circle, centred at a vertex of the Voronoi diagram and which passes through its three forming points results in the deletion of that vertex. This follows from the 'in-circle' definition of the Voronoi construction.
4. Find the forming points of all the deleted Voronoi vertices. These are the contiguous points to the new point.
5. Determine the Voronoi vertices which have not themselves been deleted which are neighbours to the deleted vertices. This information provides the necessary information to enable valid combinations of the contiguous points to be constructed.
6. Determine the forming points of the new Voronoi vertices. The forming points of new vertices must include the new point along with the two points which are contiguous to the new point and form a side of a neighbouring element. (These are the possible combinations obtained from step 5)
7. Determine the neighbouring Voronoi vertices to the new Voronoi vertices. Following step 6, the forming points of all new vertices have been computed. For each new vertex, perform a search through the forming points of the neighbouring vertices (as found in step 5) to identify common pairs of forming points. When a common combination occurs, then the two associated vertices are neighbours of the Voronoi diagram.
8. Re-order the Voronoi diagram data structure, overwriting the entries of the deleted vertices.
9. Repeat steps (2-8) for the next point.

The Delaunay mesh is generated in an automatic manner, where the use of successive point refinement is applied to generate the final mesh. Firstly the boundary points are placed into the mesh producing a valid Delaunay mesh. This mesh is then refined [32-35]. Here an example of the procedure used to add interior grid points via the Delaunay triangulation will be shown. The method is applicable to two and three dimensions, the concept being used in the example will be described for applications in two dimensions.

It is assumed that the computational domain is defined in discrete form by the boundary points. The point distribution for every point will reflect the geometrical features, such as variations in curvature and gradient. The algorithm shown will create points within the domain which reflect the point distribution function.

1) For every boundary point $r_i = (x, y)$ compute the point distribution function,

$$dp_i = 0.5(\sqrt{(r_{i+1} - r_i)^2} + \sqrt{(r_i - r_{i-1})^2})$$

where it is assumed that points $i+1$ and $i-1$ are contiguous to i.

2) Generate the Delaunay triangulation of all boundary points.
3) Initialize the number of interior nodes created as N=0.
4) For all elements within the domain,
    a) Define a prospective point, Q, to be at the centroid of the element.
    b) Derive the distribution function, dp, for the point Q, by interpolating the point distribution function from the nodes of the element, dp_m, *m=1, 2, 3.*
    c) Compute the distances d_m, *m=1, 2, 3* from the chosen point, Q, to each of the vertices of the element.

        If $\{d_m < \alpha dp_m\}$ for any *m=1, 2, 3* then reject the point:- return to the beginning of step (4).

        If $\{d_m > \alpha dp_m\}$ for any *m=1, 2, 3* then

        Store the elements containing the points. This information will be used during the insertion of each point. If the element is broken by the insertion of another point

then the point which is inside the broken element is then rejected.

d) Assign the interpolated value of the point distribution function dp, to the new node P_N.

e) Take next triangle.

5) If N=0 go to step (7).

6) Perform Delaunay triangulation of the derived points, P_j, *j=1,N*. Go to step (3).

7) Smooth the mesh.

The coefficient α controls the grid point density. The approach used here places the nodes at the centroids of the elements. Many different approaches have been used, including placing nodes along edges and circumcentres.

Figure 2.15 shows an overview of the Delaunay triangulation procedure. Where Figure 2.15 (a) shows the convex hull which encloses the boundary points. The boundary points are then triangulated, as shown in Figure 2.15 (b). The interior nodes are then added in a sequential manner, Figure 2.15 (c). All the elements situated outside of the domain of interest are deleted leaving the final triangulation, Figure 2.15 (d).

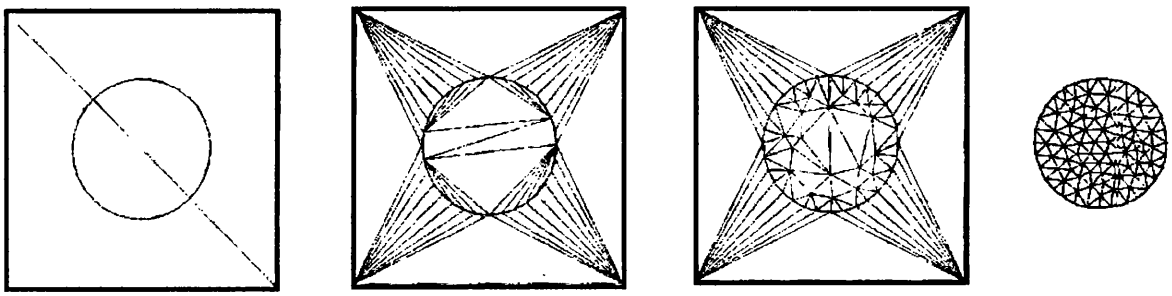


Figure 2.15 An overview of the Delaunay mesh generation process.

2.5 Mesh Quality Enhancement

In order to enhance the quality of the generated mesh, two post processing techniques are used. These processes are local in nature and do not alter the total number of vertices.

2.5.1 Diagonal Swapping

This technique is applied in two dimensions and changes the connectivities of nodes without changing their position. This process requires looping over all of the edges in the mesh, excluding boundary edges. For each side AB (Figure 2.16), common to the triangles ABC and ADB, one considers the possibility of swapping AB by CD. This replaces the two triangles ABC and ABD by the triangles ADC and BCD. The requirement that states if a swap is to take place, is if the minimum angle occurring in the new configuration is larger than that in the original one. This process works well and will increase the quality of any given mesh.

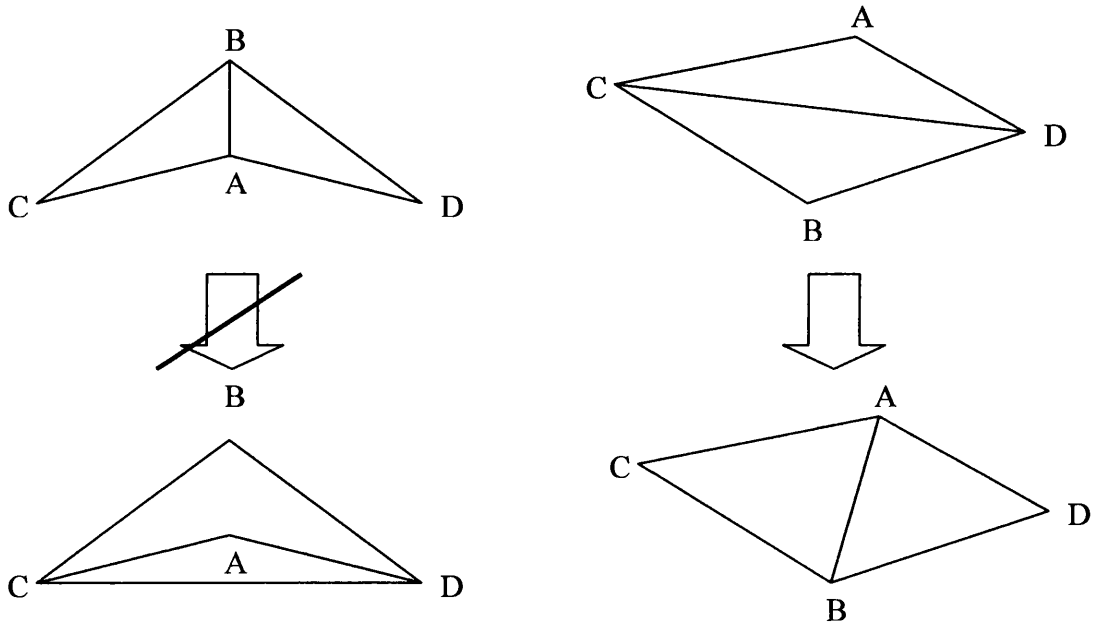


Figure 2.16 Diagonal swapping procedure.

In Figure 2.16 it can be seen that the example on the left would not be performed as the new elements would produce smaller angles than that of the previous configuration. Whereas, the example on the right would be allowed, as the new configuration would produce larger internal angles.

2.5.2 Mesh Smoothing

This procedure alters the position of the node without changing the topology of the mesh. The idea used is that every edge is considered as springs of stiffness proportional to the length of the side. The node considered is moved until the spring system is in equilibrium. The equilibrium positions are found by iteration, with each iteration consisting of looping over the interior points and moving them to the centroid of the neighbouring points. The final mesh can be smoothed as many times as the user wishes, but normally three to five iterations are performed to achieve a good representation.

The smoothing iteration looks like:

$$r_0^{n+1} = r_0^n + \frac{\omega}{M}\sum_{i+1}^{M}(r_0 - r_i) \qquad (2.6.1)$$

where r_0^{n+1} is the new position of the point r_0 after n+1 iterations, M is the number of neighbouring points with coordinates r_i and ω is the relaxation parameter. Three to five iterations are performed with the relaxation factor set to be equal to one, then reduced in the case where the minimum volume produced by moving the point to its new position is smaller than it was before.

2.6 Data Structures

The need for containing data in an efficient manner has lead to the use of data structures that allow fast storing and searching of items. Most of the operations that we will encounter in mesh generation will be that of the selection of a geometrical item: a point, side, face or element, from a set according to a certain criterion, usually proximity to another item. Another aspect of data structures that we will encounter will be that of storing information on a list that can be continuously updated by inserting or deleting items. This is particularly useful for methods such as the Advancing Front.

The two types of data structures that will allow these processes to take place will be Alternate Digital Trees (ADT) and linked lists. The following sections will introduce the two methods and describe how they will be used in mesh generation.

2.6.1 Linked Lists

A linked list is a data structure in which the objects are arranged in a linear order. Unlike an array, in which the linear order is determined by the array indices, the order in a linked list is determined by a pointer in each object. Linked lists provide a simple, flexible representation for dynamic sets.

It is easy to delete an item from within a list and to insert an item into the midst of a linked list, but there is additional memory space to be considered for links. Also, to reference a random part of the linked list will take *k* iterations to find the right place, whereas a sequential list is much faster, where to access the *k*th item in the list takes a fixed time.

A linked list can come in a number of different forms. It can be either singly linked or doubly linked, can be either sorted or not and it can be circular or not. If a list is singly linked then the link or pointer to the previous node is removed. If the list is sorted, then the order is related to the linear order of keys sorted in elements of the list. In a circular list, the previous link or pointer of the first node of the list points to the last node, and the next link or pointer of the tail of the list points to the head. Figure 2.17 shows a graphical illustration of a singly, doubly or circular linked list.

The use of linked lists in mesh generation may often require the use of an ordered list. The ordering is in increasing or decreasing order on the key field. The algorithms described below are based on an ordered doubly linked list.

Each element of a doubly linked list L is an object with a key field KEY and two other fields: NEXT and PREV. The object may also contain other satellite data. Consider an element x in the list, where NEXT(x) relates to its successor in the list and PREV(x) relates to its predecessor. If PREV(x)=0, then the element is the first in the list. If NEXT(x)=0, then the element is the last in the list. A variable HEAD is also required to point to the first element of the list. If this is equal to zero then the list is empty.

Whenever an element is to be inserted we have to check the availability list, which is a list of free elements maintained in conjunction with linked allocation. If an element is to be deleted then its position is inserted into the availability list. The attribute AVAIL specifies the top available

element of the linked list. The availability stack only contains a finite number of elements. If an element is available, then the new top-most element of the stack is denoted as NEXT(AVAIL). The key field can now be filled and the fields NEXT and PREV are set to values which designate the successor and predecessor element of the new element. A similar procedure can be formulated for the return of a discarded element to the availability stack. If the location of this discarded element is given by LOC then the NEXT field of this element is set to the present and the AVAIL takes the value of LOC.

Three important operations are often performed on a linked list.

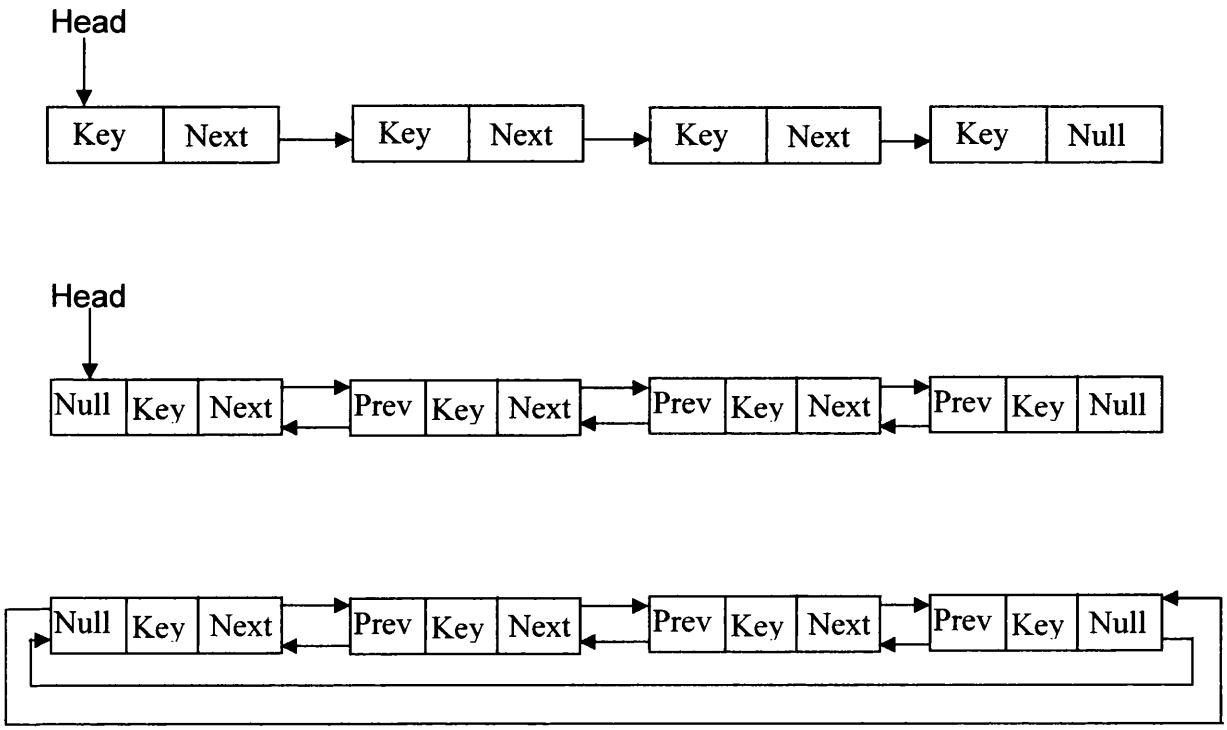


Figure 2.17 Singly, doubly and circular linked lists.

Algorithm List_Search.

To find the first element with key x in the list by a simple search, returning the location of P of the key x. If no object is found then a value of 0 is returned.

1. Set P=HEAD
2. If P=0 then return
3. If KEY(P)=x, then return
4. Set P=NEXT(P) and go to step 2.

To search a list of *n* objects, the List_Search procedure takes *0(n)* time in the worst case, since it may have to search the entire list. The algorithm starts from the first location and marches through the list until the desired element is found.

Algorithm List_Insert

This algorithm inserts an element x into a linked list preserving the decreasing order of field key.

1. If AVAIL=0, then stop as the list is not big enough
2. Set NEW=AVAIL and set AVAIL=NEXT(AVAIL)
3. Set KEY(NEW)=x
4. If HEAD=0, then
   Set NEXT(NEW)=0
   PREV(NEW)=0
   HEAD=NEW
   Exit
5. If KEY(HEAD)<KEY(NEW), then
   Set NEXT(NEW)=HEAD
   Set PREV(HEAD)=NEW
   Set PREV(NEW)=0
   HEAD=NEW
   Exit
6. Set P=HEAD
7. Repeat while P≠ZERO and KEY(NEXT(P))>KEY(NEW):
   Set P=NEXT(P)
8. Set NEXT(NEW)=NEXT(P)
9. Set PREV(NEXT(P))=NEW

10. Set PREV(P)=NEW
11. Set PREV(NEW)=P

This algorithm considers in step 1 the case where the stack is full. In steps 2 and 3 the location of the next available element is found. In step 4 we consider the case where the list is empty, which causes the links of the new element to equal zero. Step 5 considers the case where the new element precedes the first element in the original list. This will result in the new element becoming the first element in the updated list. The final case searches the original linked list to find the location to insert the new element without losing the order of the linked list. When the location is found, the links of the new element are set and the appropriate links of the previous and next elements are exchanged.

Algorithm List_Delete

This looks to delete an element x from the list. Firstly, it is necessary to call the algorithm List_Search to find the location P of the element x to be deleted.

1. Find the location P of the element to be deleted
2. If PREV(P) $\neq 0$ then
   Set NEXT(PREV(P))=NEXT(P)
   Else
   Set HEAD=NEXT(P)
3. If NEXT(P) $\neq 0$ then
   Set PREV(NEXT(P))=PREV(P)
4. Set NEXT(P)=AVAIL
5. Set AVAIL=P

Firstly the location P of the element is found, then reconnect the NEXT and PREV fields of the successor and predecessor of the removed elements. Then finally add the removed elements to the availability stack.

A linked list can improve the efficiency of the mesh generation procedure. This is shown by considering the process of updating the front

during the Advancing Front procedure. In order to obtain a high quality mesh whilst using the Advancing Front method, the front should be ordered according to the smallest edge length or the smallest enclosed angle between adjacent edges. The use of the linked list is advantageous as the front changes continuously and needs to be updated. Figure 2.10 shows an initial front consisting of 7 edges. When a new element is created, 1 side needs to be deleted from the list and then insert 2 new ones. The deletion requires the links NEXT and PREV of the previous and next edge to be adjusted. In addition, the location of the deleted edge needs to be added to the available list. This is achieved by making the position of the deleted edge to be at the top of the availability list and using the next location in the linked list to point to the previously available position. The insertion of the two new edges requires storing the key components in the next available position and the resetting of the links to and from the edge as shown in Figure 2.18.

2.6.2 Alternate Digital Trees

The Alternate Digital Tree (ADT) allows a data set to be stored in a manner that allows the insertion and deletion of data quickly; it also lends itself to a fast efficient manner of searching amongst the data. The high degree of flexibility offered by these structures makes them especially suited for applications that combine searching and sorting using both fixed and variable lists.

The searching of data is commonly referred to in the literature [103] as range searching. The algorithm [104] in this section, presents two main advantages over other range searching algorithms [103]. Firstly, the organization of the tree does not depend on the coordinates of the data in the set, it depends only on the dimensions of the set. This results in the tree being less sensitive to the order in which the points are introduced. Secondly, it also allows the storage of segments, faces and tetrahedra as well as points.

The method described in this section is that of organizing data into a binary tree structure. Firstly the data is scaled between 0 and 1 where

$0 \le x < 1$. This is to aid with the searching, where on average it takes $0(\log_2 N)$ operations, where *N* is the number of points in the set.

Binary trees are defined in a recursive manner [102]: A binary tree structure is a finite set of nodes which is either empty, or consists of a root and two disjoint binary trees called the left and right sub-trees of the root. Searching in such a structure is accomplished by assigning to each node a discriminating key. Then, starting at the root, the tree is recursively searched.

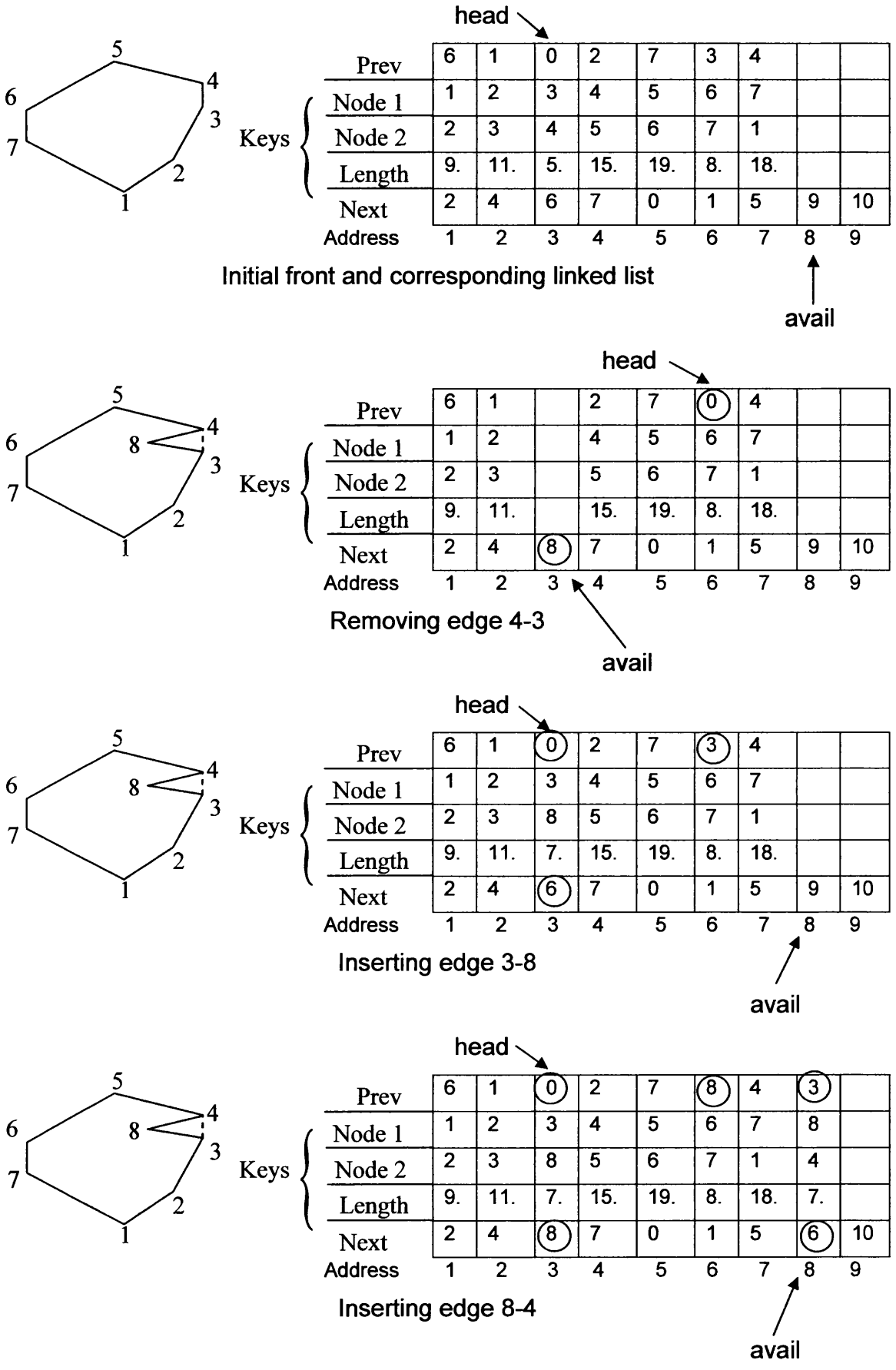


Figure 2.18 Advancing Front insertion.

When a node is visited, a query is made that either finishes the search or, according to the discriminating key, determines whether to search the right or the left sub-tree. The binary tree defined in this section starts by assigning to the root the interval $0 \le x < 1$. Each node of the binary tree is then assigned an interval $x_0 \le x < x_1$. The middle point $x_k = \frac{1}{2}(x_0 + x_1)$ is the discriminating key. The right and left sons of this node working down the tree are assigned the intervals $x_o \le x < x_k$ and $x_k \le x < x_1$, respectively see Figure 2.19.

Algorithm

The inserted points are stored at the nodes of the tree and are introduced into the tree by placing the first available point at the root of the tree. For each new point introduced into the tree, we proceed as follows:

1. Visit a node

   If the node is empty then place the point in the node and move to the next point.

   Else go to step 2.
2. The coordinate of the point to be placed in the tree is checked against the key coordinate of the node of the tree being visited.
   - If the coordinate is greater, then the key to the left sub-tree is visited.
   - If the coordinate is less than or equal to, then the key to the right sub-tree is visited.
3. Go to step 1.

For several dimensions the procedural steps are as follows. In traversing down the tree, we cycle through the N dimensions using the coordinate x_l as the discriminating value in a strictly alternating sequence. This is to say that if a node uses the coordinate x_l as its key, then its two sons will use the coordinate x_{l+1} (or x_1 if l=N) as theirs.

An example of binary tree representation of a set of points in two dimensions is depicted in Figure 2.21. This Figure also shows a case of range searching within the set to check for points within a region R where only

the nodes within the shaded path will be checked and also the path that will be taken during the searching for points close to P.

During the generation process, geometrical queries need to be performed over more general geometrical items such as, for instance, segments, triangles and tetrahedra. In the procedure presented here these items are treated as points and organised into a binary tree structure. This is accomplished by considering a box containing the item which will be defined by its maximum and minimum coordinates as shown in Figure 2.20. This box represents an interval in R_K, where k is the number of dimensions. This interval can be represented as a point in a space of 2*k dimensions.

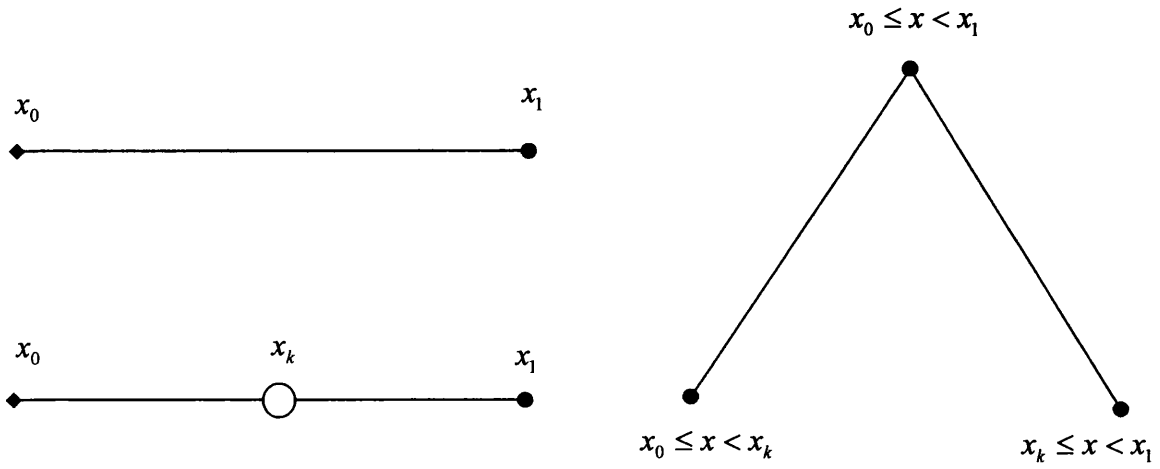


Figure 2.19 A sub-tree and its associated segments.

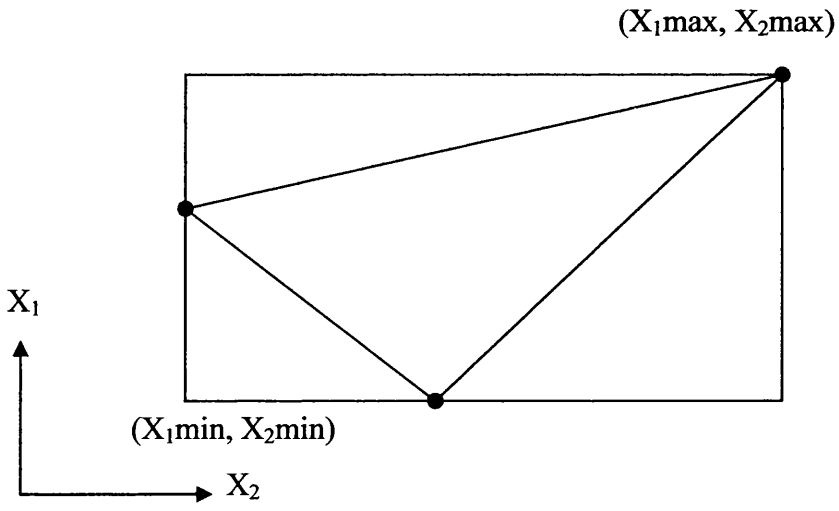


Figure 2.20 Interval associated to a geometrical item.

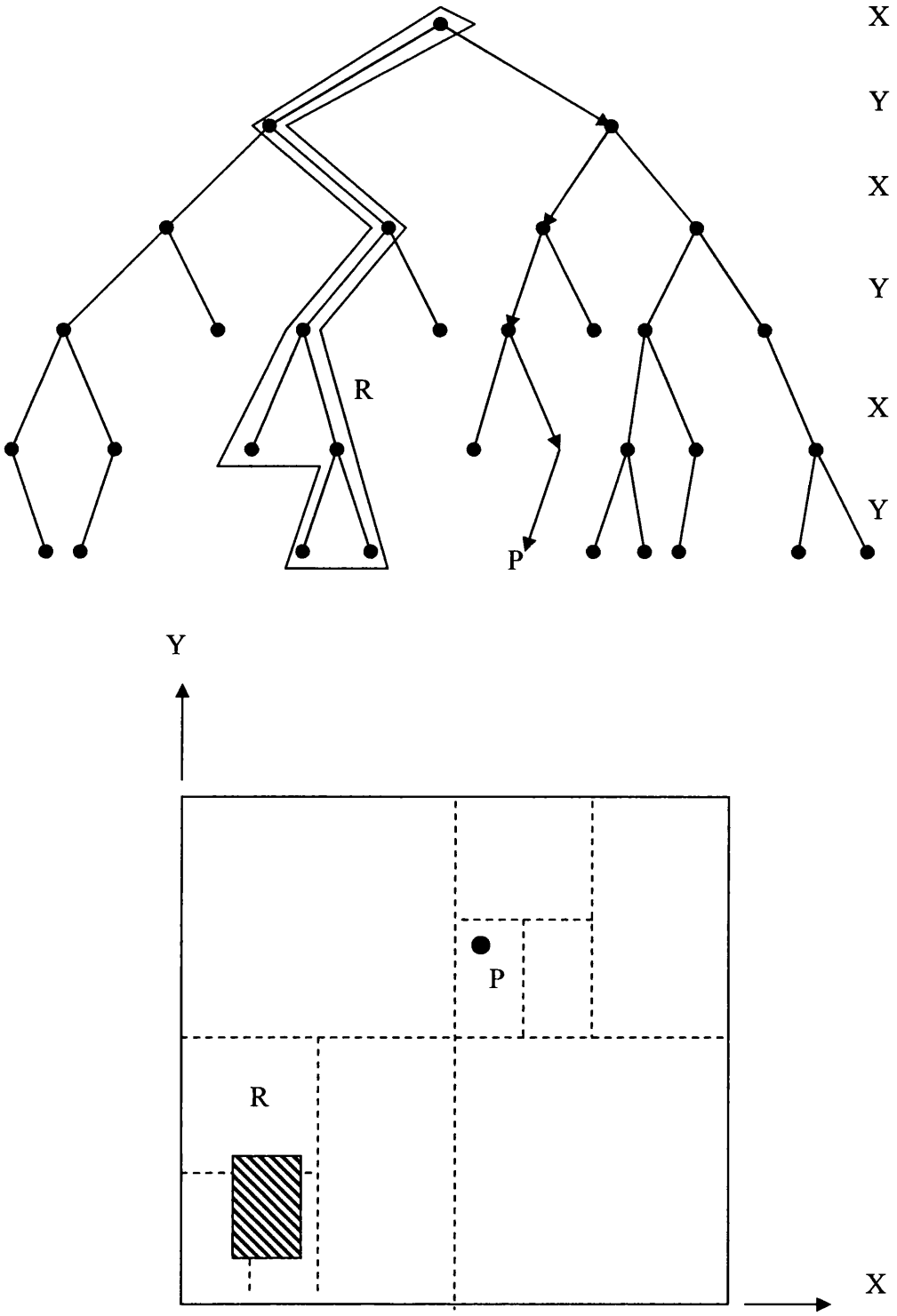


Figure 2.21 An example of two dimensional tree structure. To check for points inside the region R only the nodes in the shaded path are checked. The arrows indicate the path followed in searching for points close to P.

3.

Super Patches

3.1 Introduction.

Generating meshes from a CAD model for computational analysis is classed as the norm, but generally direct analysis on the model is difficult to find. The CAD geometries will often contain a number of detailed features which will need to be improved by processes such as CAD repair before mesh generation can take place. Even then the geometries can still contain problems in the features such as, small sliver surface patches and sliver edges. Also the surface definition can contain gaps in the geometry between surfaces or curves. These features cause major difficulties when generating meshes.

The CAD models that are being generated by designers are becoming more and more complex, which in turn causes more problems when reproducing the CAD model for mesh generation. The models used often contain a number of different details for aesthetic and manufacturing purposes only which are of no relevance to the analyst. Here we look to eliminate these small features by the use of the Super Patch method.

Previous researchers have looked at repairing the geometry and trying to simplify the CAD geometry to enable mesh generation to take place. Such work by researchers such as Butlin [24, 25], have focused on repairing the geometry by regenerating the geometrical and topological representations of the surface definition if it is not classed as suitable for mesh generation. This process will resolve some of the problems mentioned above, but will not eliminate the small slivers and narrow surfaces that are generated by the CAD designer. Other researchers have looked at less computationally expensive

methods such as Sheffer [26,27] where an additional layer of new topology is generated on top of the original model to simplify the model's topological definitions. More recent work by Lee [28] looks at the introduction of the medial axis to simplify the CAD definition and to eliminate slivers. The method merges together neighbouring surface patches of similar tangency, to obtain one Super Patch that can be meshed as a single surface.

The method proposed assumes that the geometry has undergone some form of CAD repair to ensure that the geometry is water tight. The features that the CAD repair could not eliminate will be looked at in this method, these include (Figure 3.1 and 3.2):

- Slivers - this is either, the surface patches are small or the edges used to represent the geometry are small.
- Narrow surface patches - this is a surface patch where two of the boundary lines come in close contact.
- Badly shaped patches.

These features cause more elements to be required to represent the model than is necessary.

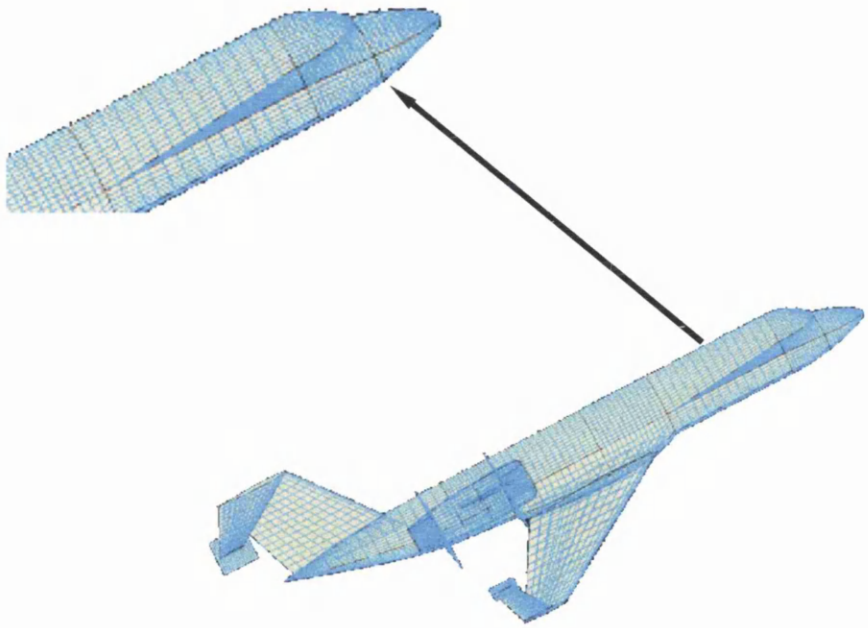


Figure 3.1 CAD definition of Gulf-stream showing triangular shaped patch, which as been used for aesthetic purposes only.

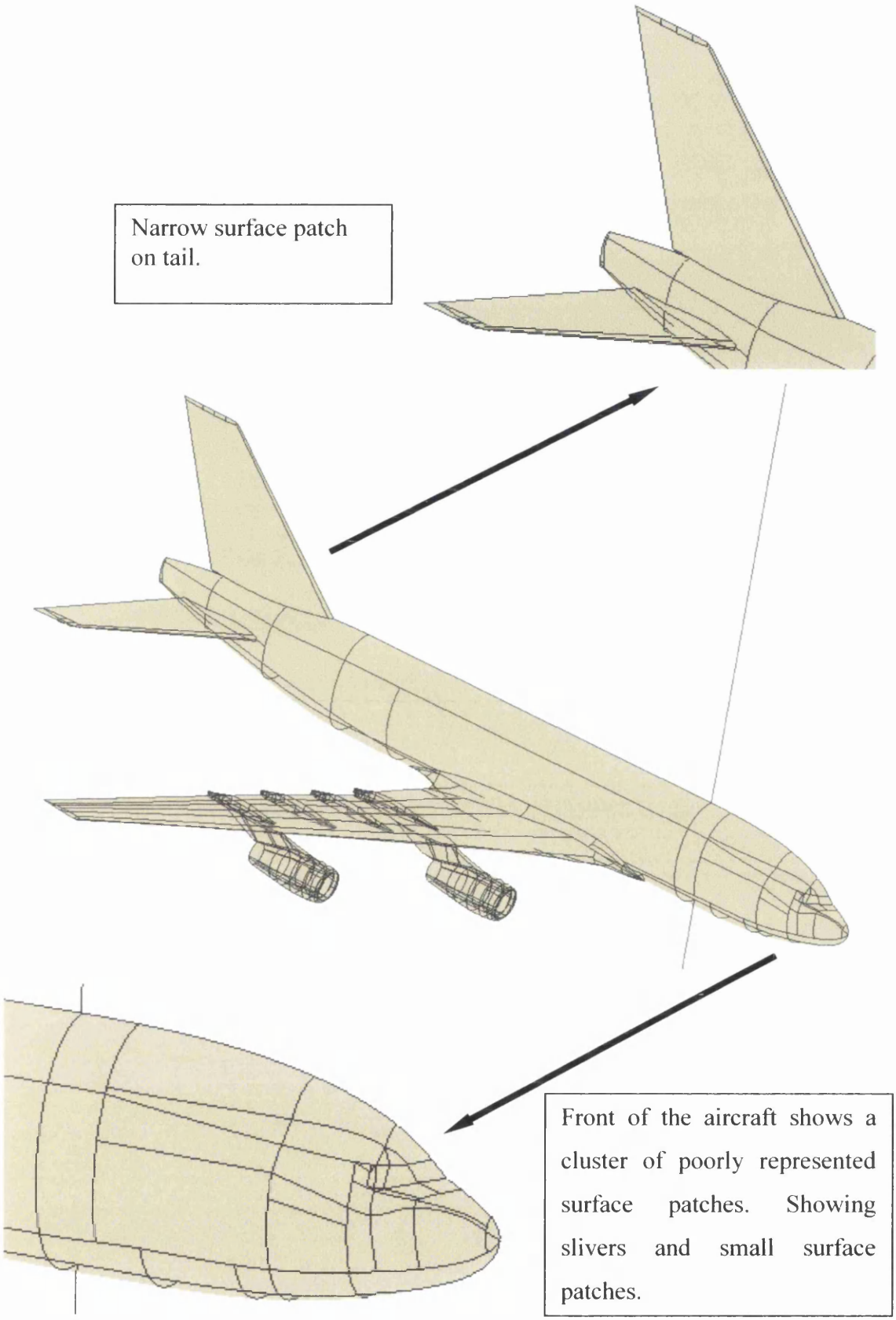


Figure 3.2 Surface patch representation.

3.2 Super Patch Algorithm

In this chapter the Super Patch Algorithm will be introduced to compute the Super Patch method. This method will be able to :-

1. Generate a coarse mesh on each individual surface patch.
2. Generate Super Patches. Identify surfaces to be merged together to form a Super Patch. Eliminate duplicate curves, from the Super Patch leaving only boundary curves.
3. For each Super Patch:
    a. Transfer the local parametric coordinates for each individual surface patch in the Super Patch to a global parametric coordinate system, to enable the mesh generation process to take place.
    b. Remove duplicate points, which have been created from the merger of the surface patches.
    c. Generate the initial front for the Super Patch.
    d. Move the initial front by a fraction of the inward normal if needed.
    e. Compute the triangulation on the Super Patch by the Advancing Front method.
    f. Apply mesh improvement techniques, such as swapping diagonal and node smoothing.
4. If there are no Super Patches left then finish, else go to 3.

3.2.1 Generate Coarse Mesh

Due to the fact that all the surface information available is based on each individual surface patch, then information will have to be transferred back, to be able to place points onto the 3D surface. Therefore, the first task is to gather together the information on each individual surface patch to enable the generation of the Super Patch to take place. This is done by generating a coarse mesh on each of the individual surfaces, using the normal Advancing Front method as described in Chapter 2, and storing the surfaces three-dimensional coordinates, parametric coordinates and the connectivity array. Along with the connectivity array, the elements surface

number is stored to enable easy location of surface identification. The reason for storing this information is because the individual surfaces are generated in the local normalised space, hence when generating on the Super Patch there's a need to refer back to the local normalized space to be able to calculate the three dimensional coordinates of any given point from its corresponding surface number.

3.2.2 The Creation of Super Patches.

After the generation of a coarse mesh on each individual surface, the next step in the process is to determine which surface patches can be merged together to create Super Patches. Each Super Patch is created using surfaces of similar tangency. The reason for this is to be able to eliminate the problems mentioned in section 3.1 whilst still retaining an accurate representation of the surface geometry.

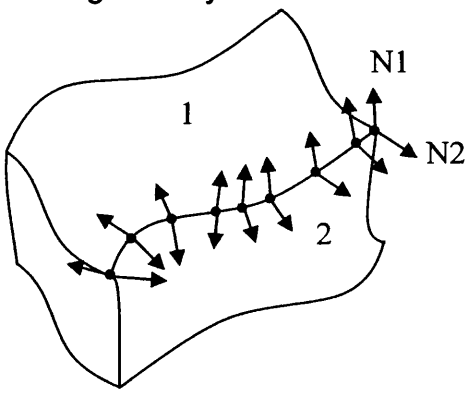


Figure 3.3 Calculation of normals for every point along the curve relating to the two adjacent surfaces.

Algorithm

- Select a starting surface.
- Create a list of the boundary curves surrounding the surface.
- For every point on the boundary curve compute the normals associated with the two adjacent surfaces, Figure 3.3.
- If the maximum angle between the normals is less than a given tolerance, set at one degree, then:
    - Remove the curve from the list and add the adjacent surface to the Super Patch list. This is done using a linked list, in

which the objects are arranged in a linear order. The order in the linked list is determined by a pointer in each object.

- Add the boundary curves of the adjacent surface to the list of curves to be checked.

- Continue checking surfaces until no surfaces remain and the domain has been replaced with Super Surfaces.

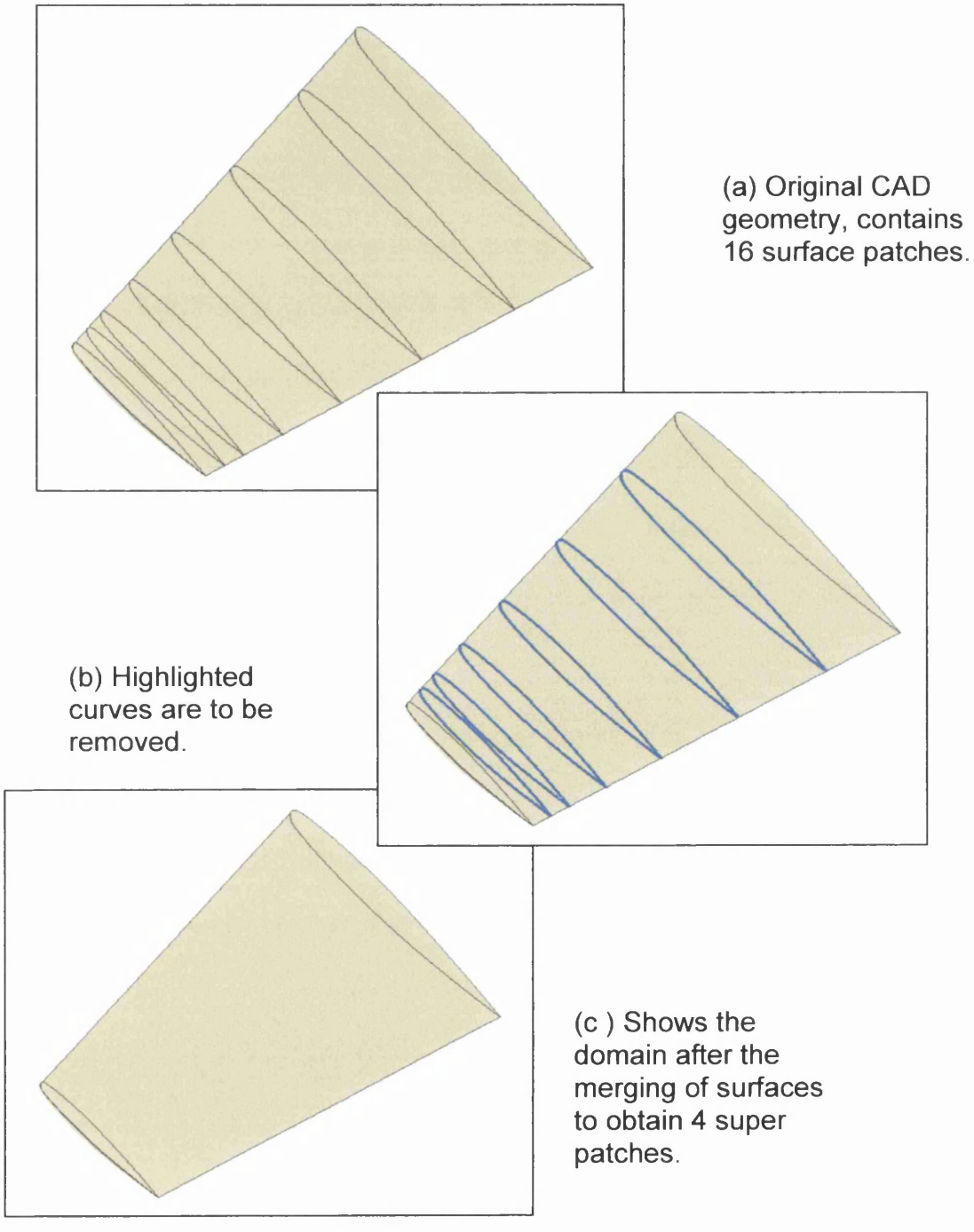


Figure 3.4 Example of merging surfaces for the M6-wing.

An example of this process can be seen in Figure 3.4 where the m6-wing has been used to demonstrate the merging of surface patches. The original CAD geometry contained 16 surface patches, Figure 3.4a. Once this process of merging surface patches together has been applied to the geometry the curves highlighted in Figure 3.4b were removed and the adjacent surfaces were merged together to form 5 Super Patches to define the domain, Figure 3.4c.

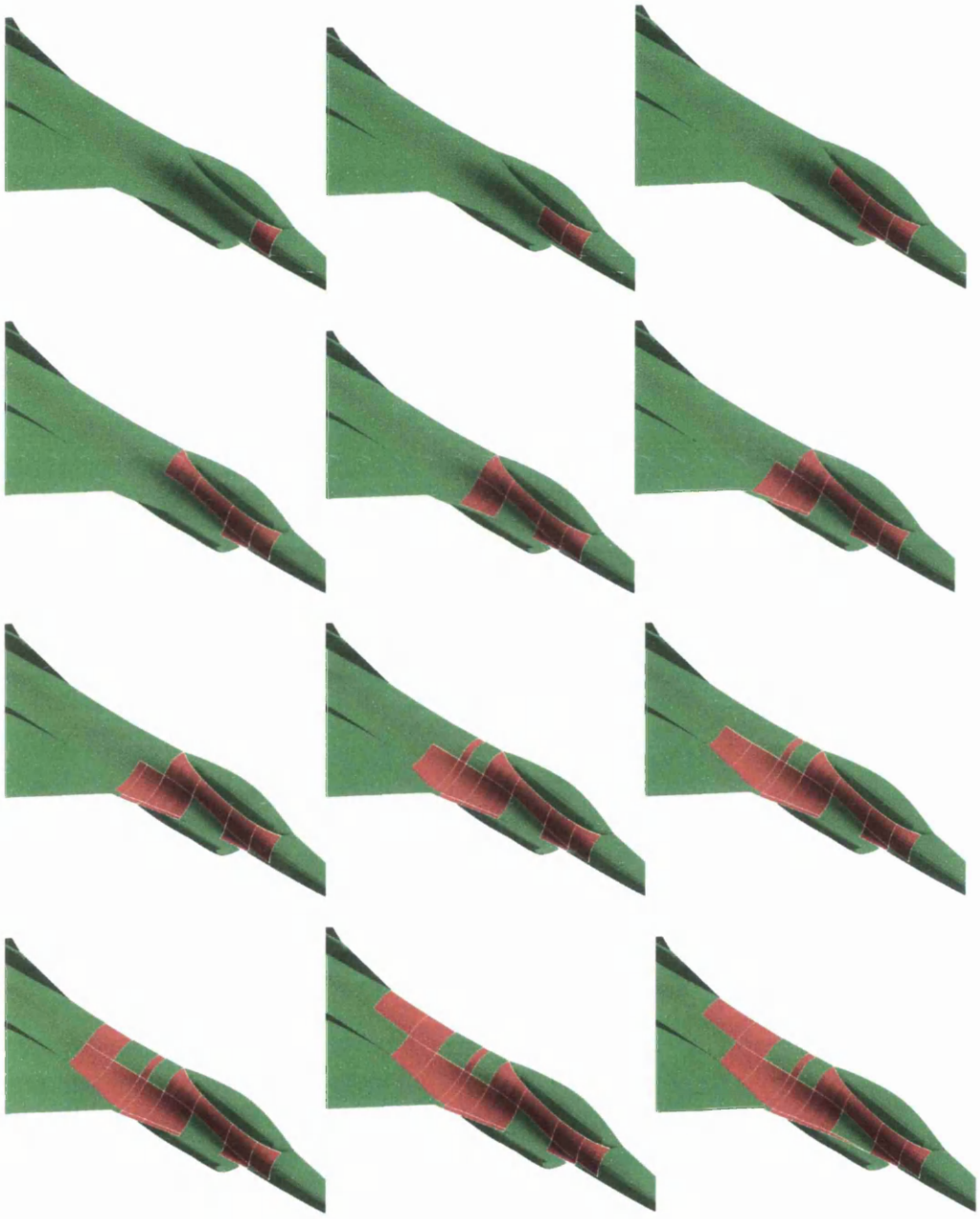


Figure 3.5 Shows the automatic choosing for one Super Surface on the F16. Sixteen individual surfaces make up one Super Patch.

Another example of this process can be seen in Figure 3.5 where the F16 fighter plane has been used to show each stage in the generation of an individual Super Patch.

3.2.3 Transformation from Local to Global System

The way in which the CAD geometry is defined allows every surface patch in the formation of the domain to have a different surface definition i.e. the *u, v* coordinate system in the parametric domain can be of varying directions from patch to patch, Figure 3.6.

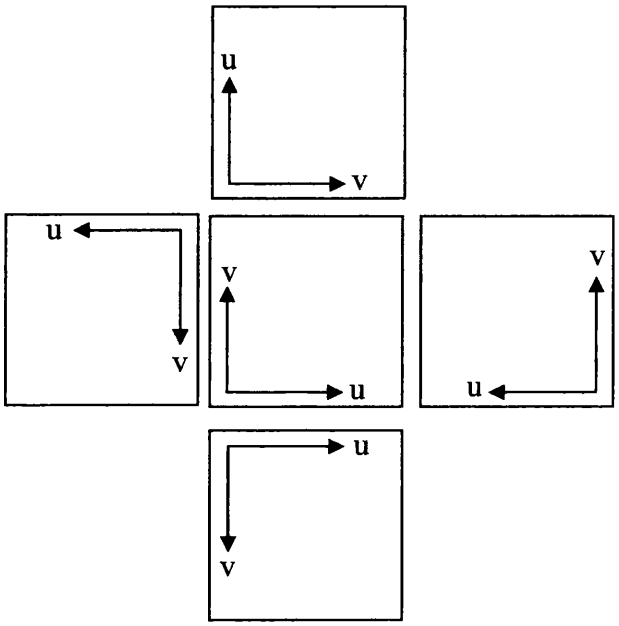


Figure 3.6 Shows how a surface patch can be surrounded by a number of surfaces with different surface definitions to itself.

This is acceptable for the original mesh generation process described in Chapter 2, where each surface patch is generated individually. However, in this chapter neighboring surface patches will be merged together and meshed as a whole, making sure that the *u, v* coordinate system used is constant throughout the Super Patch.

A number of different coordinate systems occurred, which meant that every possible case had to be thought of and catered for. There were 32 possible cases in all, some of which can be seen in Figure 3.7.

Each possible situation had to be looked at and a method devised that would cater for the transfer of the coordinate systems to one global coordinate system. The method used involved the transfer of every surface to relate to the first surface in the list. A number of procedures have to be applied to obtain a global representation of the geometry; these are known as scale, shift, swap and rotate.

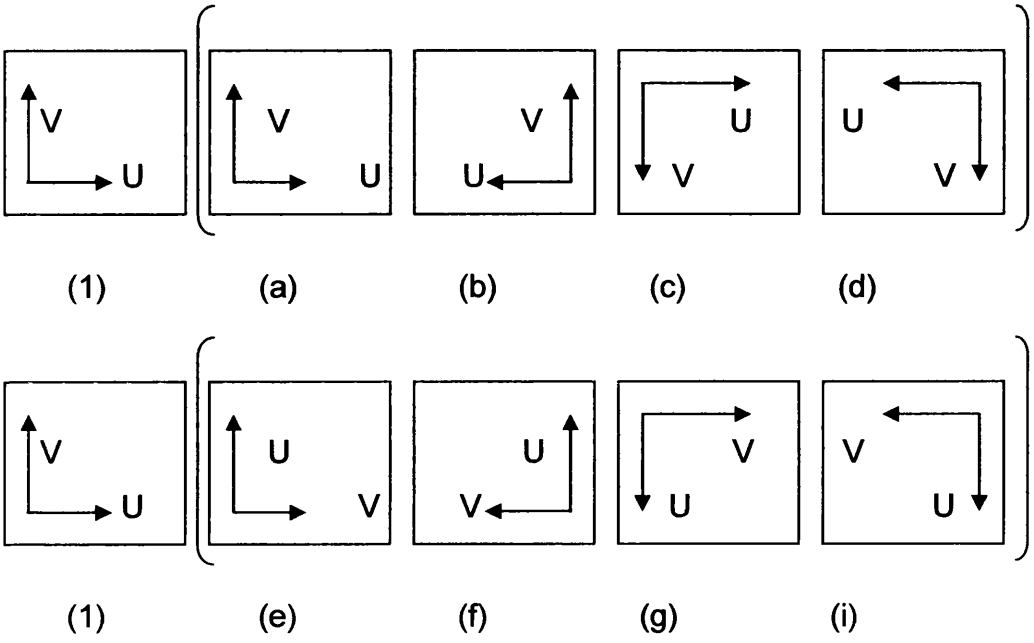


Figure 3.7 Shows the eight possible cases that can be associated with situation (1)

These processes are described as:

- *Scaling:* This is where intersection edges are scaled so that they have the same value on either side of the edge, Figure 3.8. The scaling is obtained by:

$$\frac{\gamma}{\beta} = \alpha$$

where γ and β represents the number of divisions along the edge AB on surface one and surface two respectively. Then the points along AB on surface 2 are then multiplied by α to obtain the new coordinates in relation to surface 1. Figure 3.8b and Figure 3.8c show an example of this process, where $\gamma = 10$ on surface one and $\beta = 20$ on surface two,

therefore $\alpha = 0.5$ and hence vertex B on surface two becomes $20 \times 0.5 = 10$.

- *Shifting:* The parametric system is set in the local domain, therefore, all the surfaces in the Super Patch have to be shifted by means of the shift process into a global system, where the mesh generation can take place. The global system is chosen to be the same system as that of the first surface in the list of the surfaces that make up the Super Patch. Figure 3.9 shows a global parametric coordinate system for a Super Patch on the F16 fighter plane. An example of this process can be seen in Figure 3.9 (b and c), where Figure 3.9b shows the surface patches before the shifting process. The intersecting curve joining the two surfaces is AB. It can be seen from this figure that the surfaces are placed on top of each other, due to the fact that the surfaces are represented locally. The patches both have the same system, where u=[0,10] in surface one and u=[0,15] in surface two. To produce the global system, every point along the u-direction of surface two needs to be increased by 10. If the surfaces were joined along the edge CD then there would have to be a subtraction of 10 from every point in the u-direction. The resulting coordinates are u=[0,10] for surface 1 and u=[10,25] on surface 2. Figure 3.9a shows a global representation of a Super Patch, showing how complex the surface patches can become.
- *Swapping:* Depending on the way in which each surface is declared there may be a need to swap the coordinate direction, either in the *u* or *v* plane or both depending on the surfaces. This process is used to ensure that all the u and v directions are transferred into one global [u,v] representation. Figure 3.10 shows the swap process applied to the u-direction of surface 2. It can be seen how the original u co-ordinate system is u=[0,10] for surface 1 and u[0,-10] on surface 2, after the swapping process u=[0,10]. The swap procedure does not rely on the coordinate systems being the same, it only looks at the direction of the coordinate systems.
- *Rotation:* The surfaces as shown in Figure 3.11, may have the system directions for both surfaces but the u and v are not consistent. This

means that the u and v directions have to be rotated to ensure a correct global representation. The original representation has v=[0,15] and u=[0,10]. After the rotation has taken place v=[0,10] and u=[0,15].

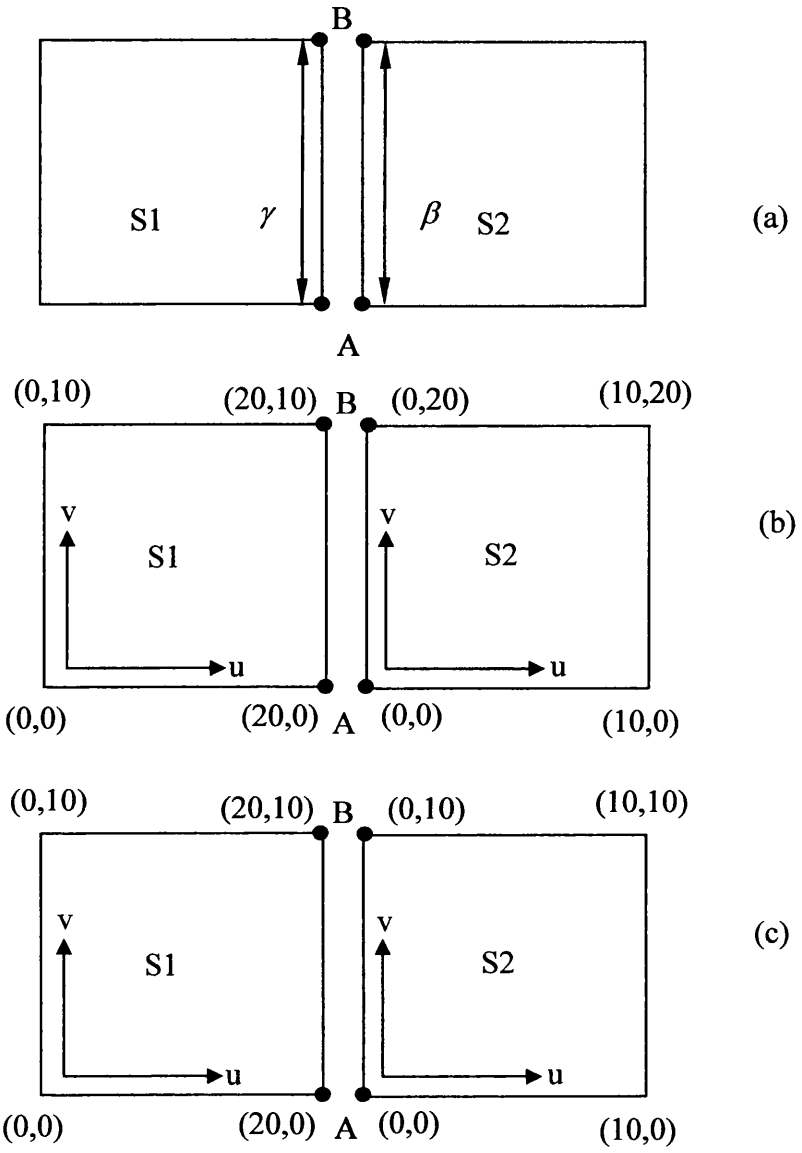
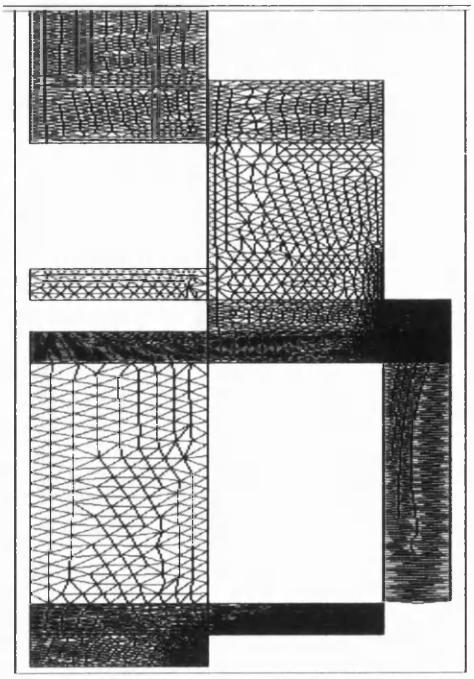
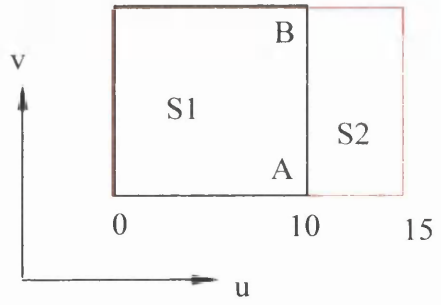


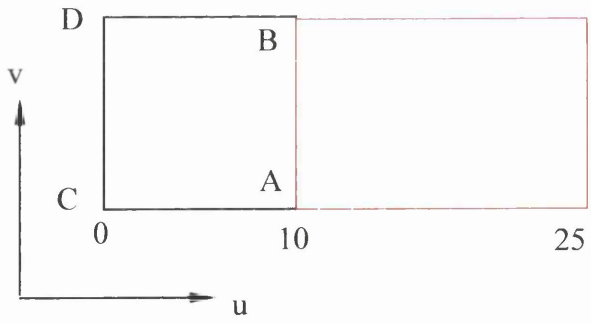
Figure 3.8 Example of the scale process.



(a) Global Parametric surface made up of 16 surfaces.



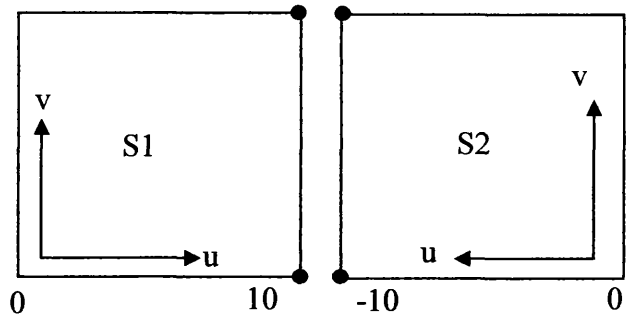
(b) Surface patches before shifting process.



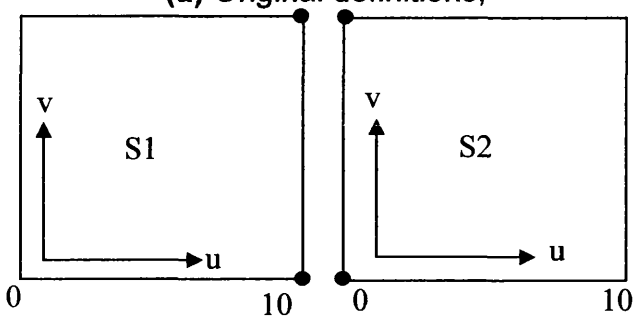
(c) Surface patches after shifting.

Figure 3.9 Example of the shifting process.

The complete process can be seen in Figure 3.12 where a flow chart is used to show the approach in which the shift, swap, scale and rotate procedures take place.

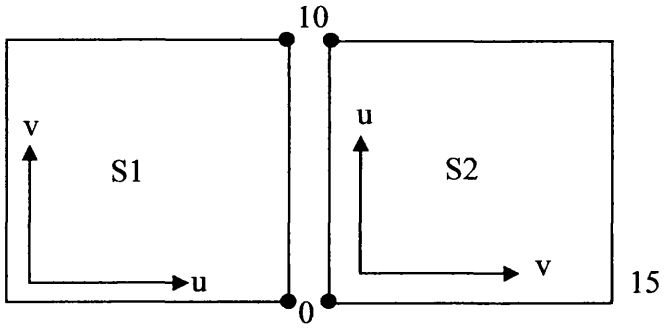


(a) Original definitions,

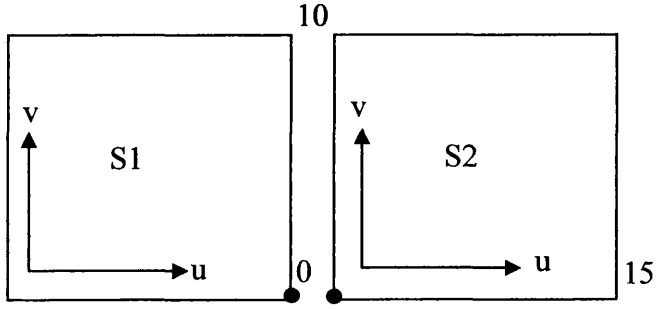


(b) Surfaces after swap.

Figure 3.10 Example of Swap procedure.



(a) Original definitions.



(b) Surfaces after rotation.

Figure 3.11 Example of rotation of directions.

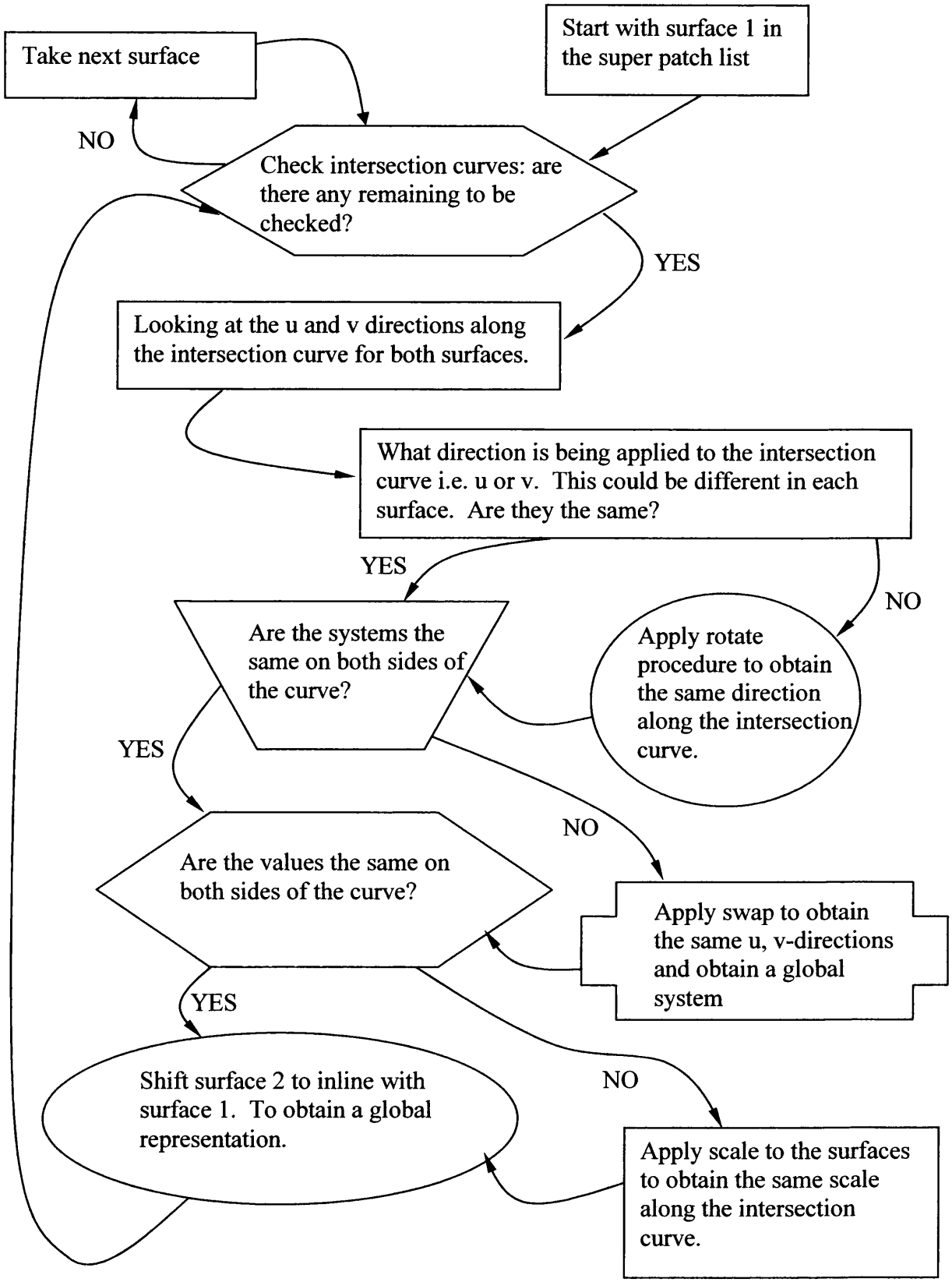
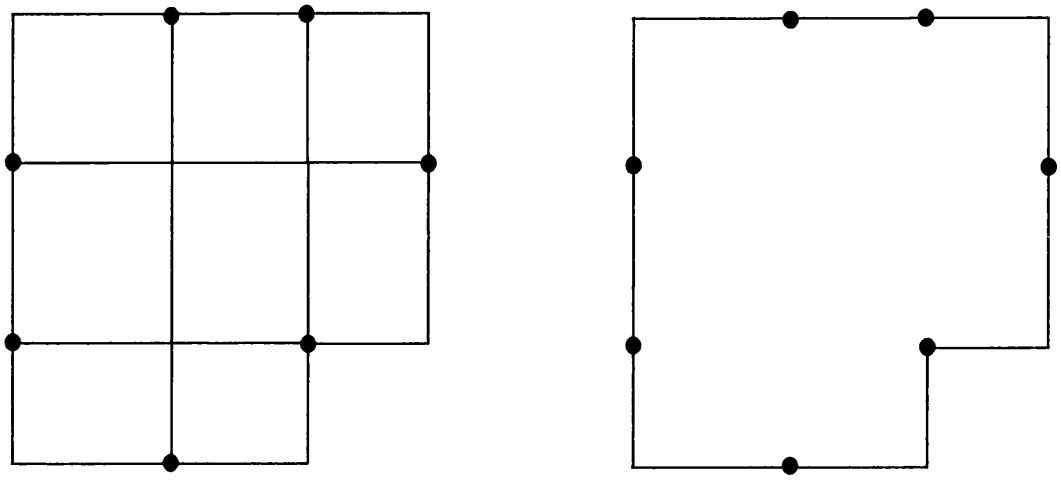


Figure 3.12 Flow chart showing the global representation procedure.

3.2.4 Duplicate Points

Once the global system has been generated, as described in section 3.3.2, duplicate points that have been created by the shift, swap, scale and rotate procedures have to be removed, Figure 3.13. These duplicate points occur along the internal boundary of the Super Patch. The points are removed by searching through the domain and if any have the same co-ordinates then the point needs to be removed from the list and all related information updated.



(a) Shows individual surface patches and duplicate boundary points.

(b) Super Patch boundary curves and duplicate points.

Figure 3.13 Duplicate points created by global system.

3.2.5 Generation of Initial Front

After all the processes described above have been implemented, all that remains are the boundary curves. This will enable the initial front to be generated in the same manner as described in Chapter 2, utilising the Super Patch boundary curves.

This process for the Super Patch method is the same as the original but care has to be taken when dealing with multiple surfaces as a situation can occur where different surface patches have the same parametric co-ordinates when they should be different. This situation can be seen for the Super Patch shown in Figure 3.14, where the global parametric co-ordinates for this surface are shown in Figure 3.15. It can be seen how different

surfaces have been given the same parametric co-ordinates when this is not the case.

Figure 3.14 Super Patch made up of 16 individual surfaces.

The reason for this is that during the global transformation, the surfaces have only been shifted vertically and not horizontally. This situation caused the resulting mesh to be produced as shown in Figure 3.16. The mesh has been generated allowing boundary curves to be connected via the outside of the domain, which is incorrect, as elements can only be created on the inside of the domain.

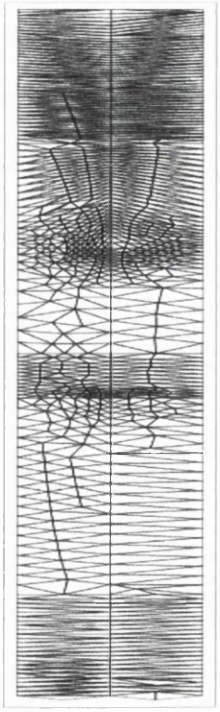


Figure 3.15 Global Parametric coordinates of Figure 3.14.

In order to overcome the problem of elements being created outside the domain, the idea of moving the parametric co-ordinates inwards by a small value in the direction of the inward normal was introduced. Two methods were looked at to implement this idea:

1. Moving every boundary point on the initial front by this value.
2. Only moving the duplicate curves by this value.

The first method was chosen. The points along the boundary were moved by:

- Choose an edge.
- Compute the inward normal of the edge.
- Multiply connecting points by a fraction of this value.

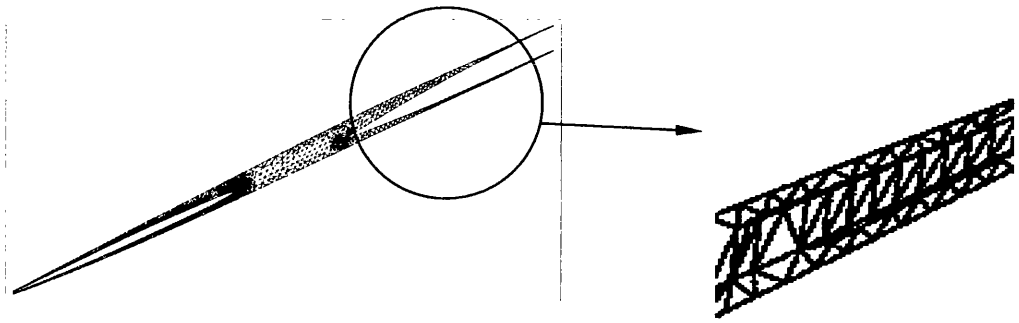
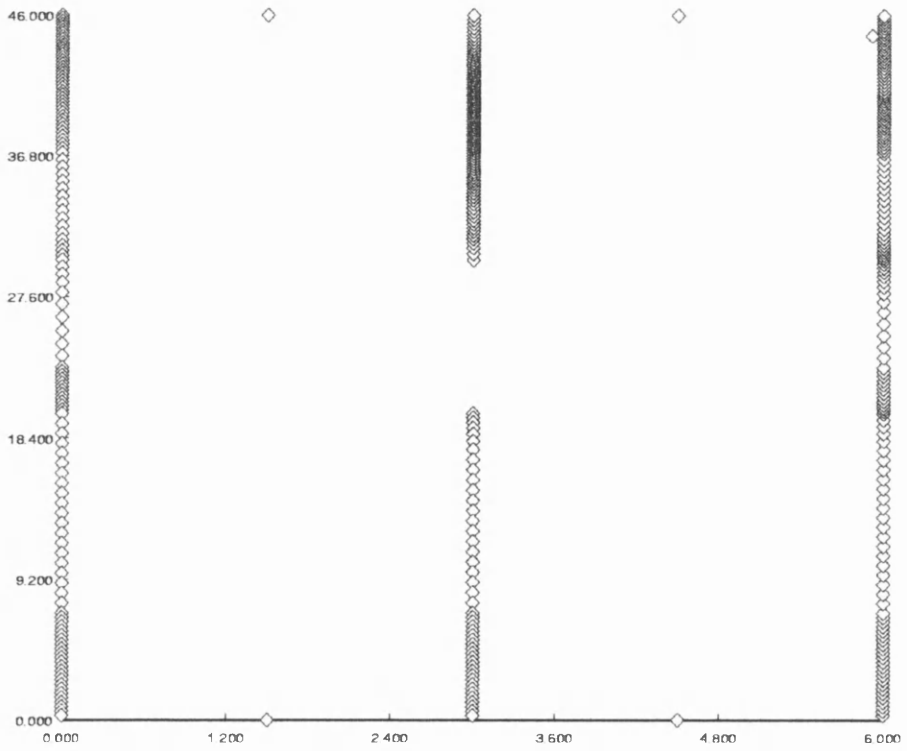


Figure 3.16 Shows a close up of the boundary curves being connected on the outside of the domain.

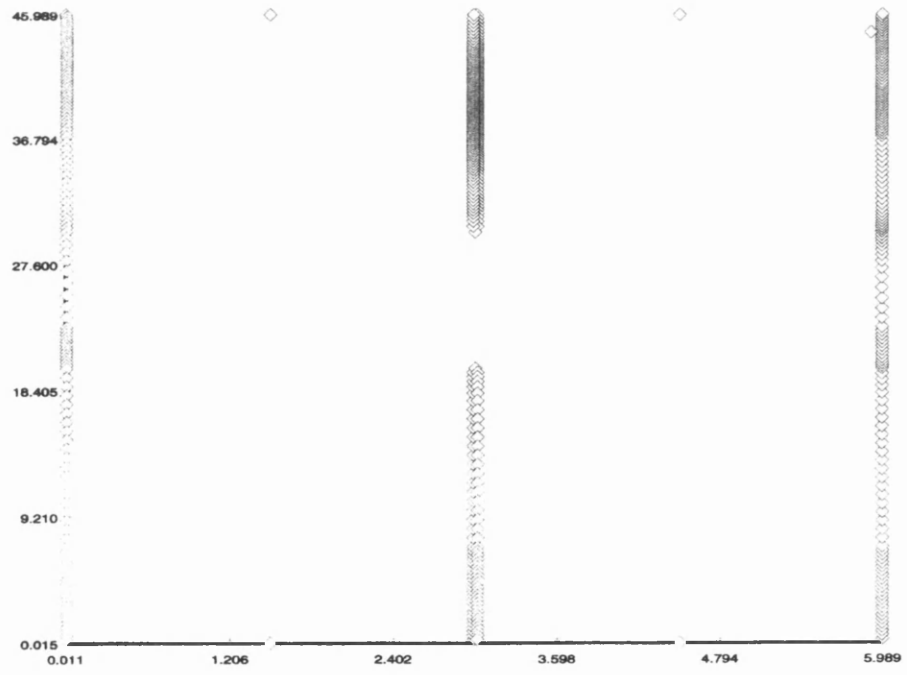
An example of this procedure can be seen in Figure 3.17 where Figure 3.17a represents the original co-ordinates, then after the points have been moved inward by a fraction of the normal the coordinates are as shown in Figure 3.17b.

3.2.6 Generating Elements

The triangle generation algorithm utilises the concept of the initial front as described previously, where the process starts with the front consisting of a sequence of straight line segments connecting consecutive boundary nodes. The process for generating elements on the Super Patch follows the same method as in Chapter 2 using the Advancing Front method.



(a) Original boundary



(b) Boundary showing the duplicate curves now having different parametric coordinates.

Figure 3.17 Boundary points before and after applying the movement of nodes by a fraction of their inward normal

The difference with this method is in the mapping. When a point has been chosen as the ideal point P the coordinates are projected from the global parametric system onto the coarse mesh generated at the start of the process, Figure 3.18. The coarse mesh allows the original surface number for the point to be found. Once found the original surface definitions are used to transfer the coordinates into physical 3D coordinates, in the same manner as described in Chapter 2.

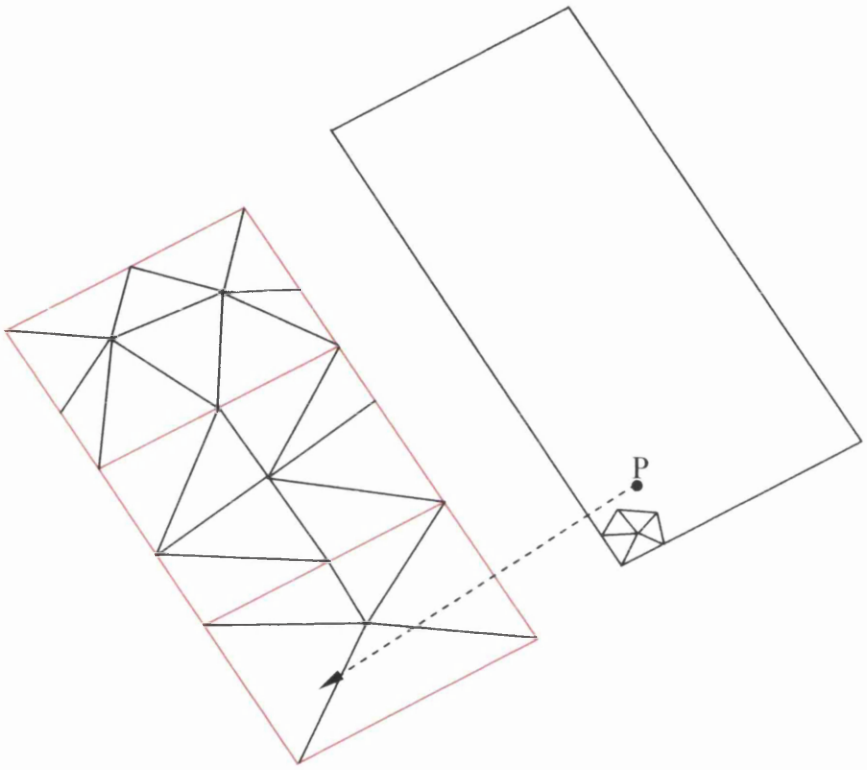


Figure 3.18 Projection of point from the global parametric system to the original coarse parametric triangulation.

The final stage in the process is to apply mesh improvement techniques. These include swap diagonal and node smoothing as described in Chapter 2.

3.3 Examples

Throughout this chapter a method of merging together surface patches of similar tangency has been introduced. The method improved the resulting

mesh quality and eliminated problems of small, narrow and badly shaped surface patches.

In order to check the quality of the new mesh produced in relation to the original surface mesh a number of quality measures have been applied. The first quality measure checks the shape by the aspect ratio of the element and is defined as follows:

$$Qual_T = \alpha \frac{h_{\max}}{\rho_T},$$

where $h_{\max}$ is the elements longest edge, ρ_T is the in-radius of the triangle T and α is a normalization coefficient chosen so that the quality of the equilateral triangle is equal to one. This function will range between 1 and ∞, where the optimal value is one. For a mesh Ω, a global measure is computed as:

$$Qual_G = \max_{T \in \Omega} Qual_T .$$

where $Qual_G$ is the worst value. Also the average quality ($Qual_{avg}$) of all the elements is calculated, along with the percentage ($Qual_{per}$) of elements having a quality better than 2 to show the overall mesh quality.

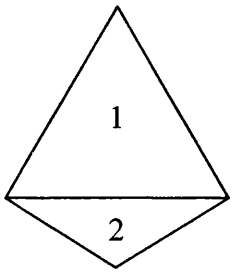
Table 3.1 shows the shape quality measures for all the following examples. It must be noted, that in some cases there may not be any improvement to the quality due to the fact that the region that incorporates the worst elements may not be merged to form a Super Surface.

Figure	No. elements	No. points	$Qual_{avg}$	$Qual_G$	$Qual_{per}$
Original gulf	62134	31053	1.53	6.58	91.3
Super gulf	62954	31586	1.46	5.80	94.1
F16 original	163806	81905	1.42	5.66	90.2
F16 Super	162913	80321	1.43	5.70	92.4
A3XX original	45414	22705	1.60	9.40	89.98
A3XX Super	45324	22688	1.54	6.41	92.6

Table 3.1 Statistics for examples

The other quality measures relate to length, area and angle. Firstly edge lengths are considered where $l_{\min}$ and $l_{\max}$ are the min and max lengths respectively. The ratios between the smallest and largest edge lengths to meet at every point are considered. The average case throughout the whole mesh is l_{ratio}, where the optimum is 1 and the worst case is infinity.

The next quality measure relates to element area, where the ratio between element areas that share an edge are computed. The area quality measure in Table 3.2 is the average, throughout the mesh, where the optimum value is 1 and the worst case is infinity.



$$\frac{l\arg est(area)(1)}{smallest(area)(2)} = area_{ratio}$$

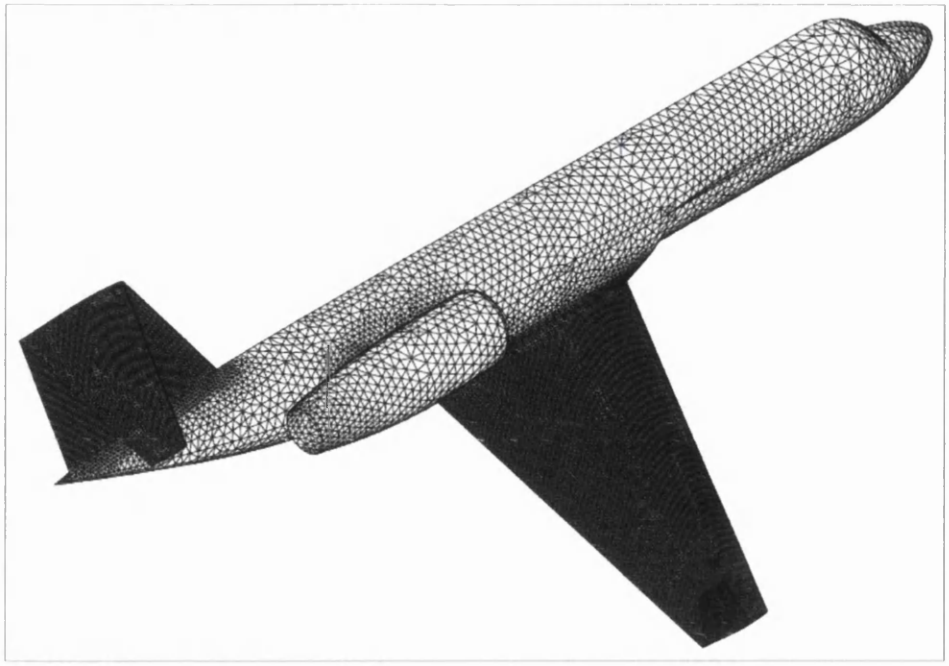
The final quality measure computes angles, where $ang_{\min}$ and $ang_{\max}$ represent the smallest and largest angle throughout the mesh, where the optimum for an equilateral triangle is 60 degrees.

Example	$l_{\min}$	$l_{\max}$	l_{ratio}	$area_{ratio}$	$ang_{\min}$	$ang_{\max}$
Gulf	0.6	0.98	1.21	1.24	10.1	91.3
Gulf new	0.6	0.98	1.20	1.25	45.1	67.7
F16	0.01	0.1	1.31	1.28	8.9	94.2
F16 new	0.01	0.1	1.30	1.278	43.7	70.9
A3XX	511.423	1532.1	1.28	1.35	7.6	145.6
A3XX new	511.423	1532.1	1.28	1.35	48.6	68.9

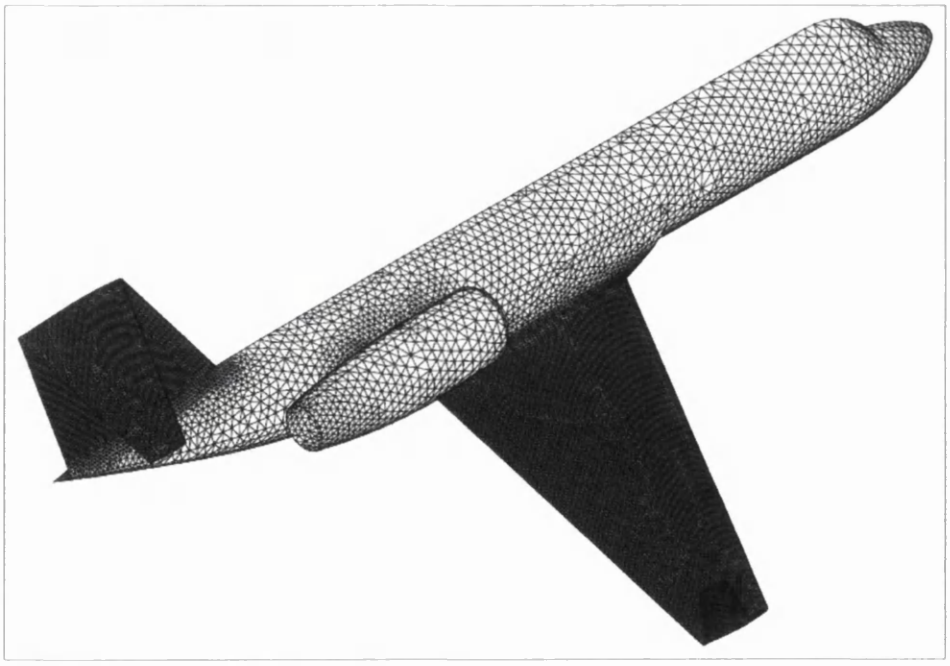
Table 3.2 Quality measures, length, area and angle.

The first example is that of the Gulf-Stream aircraft where the original surface mesh, Figure 3.19a includes a small triangular surface patch at the front of the aircraft. This has resulted in more elements being used to define the surface than is necessary. The Super Patch method has been applied to

this geometry where the resulting mesh is shown in Figure 3.19b. It shows how the surface patches in the area under consideration have been merged together and the triangular patch eliminated resulting in a more consistent element distribution. Figure 3.20 shows a close up of the area being described.

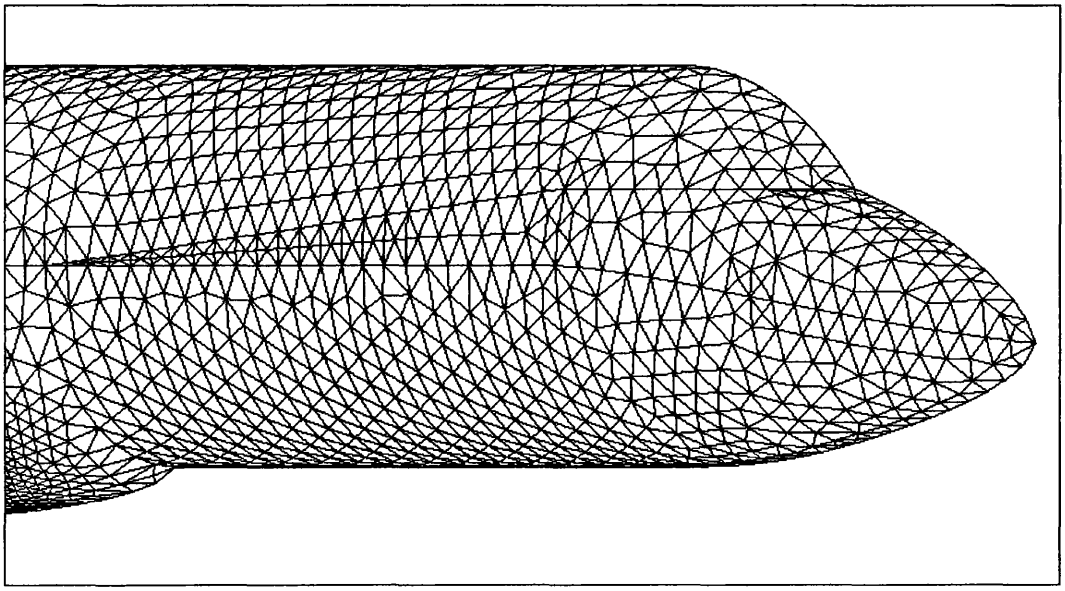


(a) Original surface mesh.

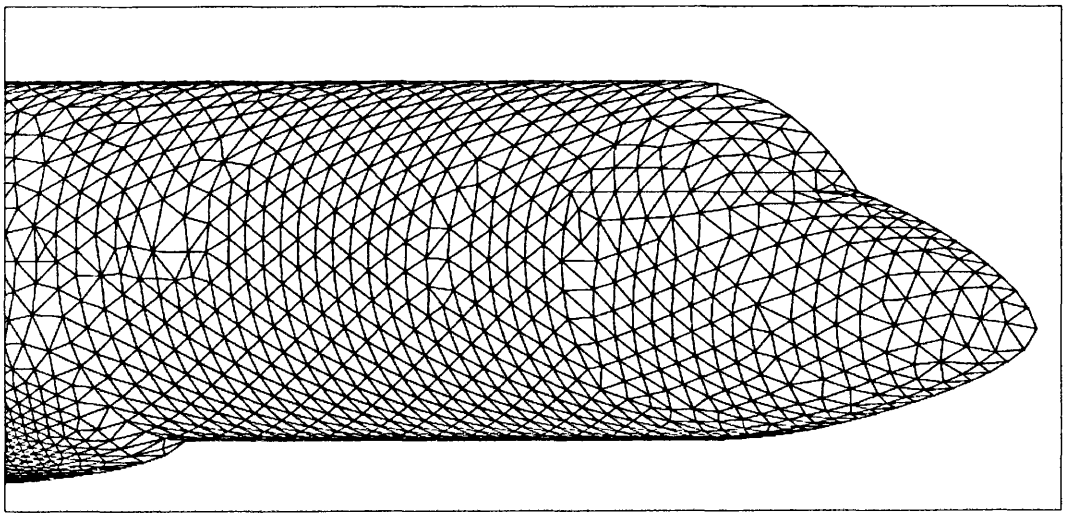


(b) Super Patch mesh.

Figure 3.19 Gulf-Stream using original and Super Patch methods



(a) Original mesh.



(b) Super Patch mesh

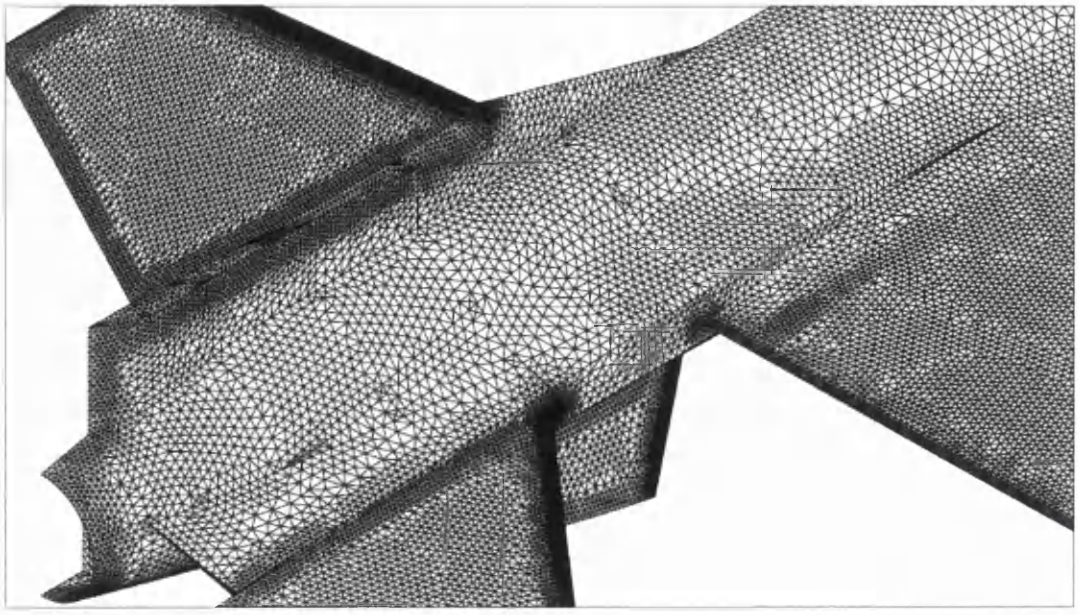
Figure 3.20 Gulf-Stream example.

The second example is that of the F16 fighter plane, Figure 3.21 where the tail of the original aircraft includes some small triangular surface patches similar to the previous example. This shows how the resulting mesh has removed the small elements produced due to this problem without losing any quality in the surface definition. Figure 3.22 shows a close up of the tail end of the aircraft.

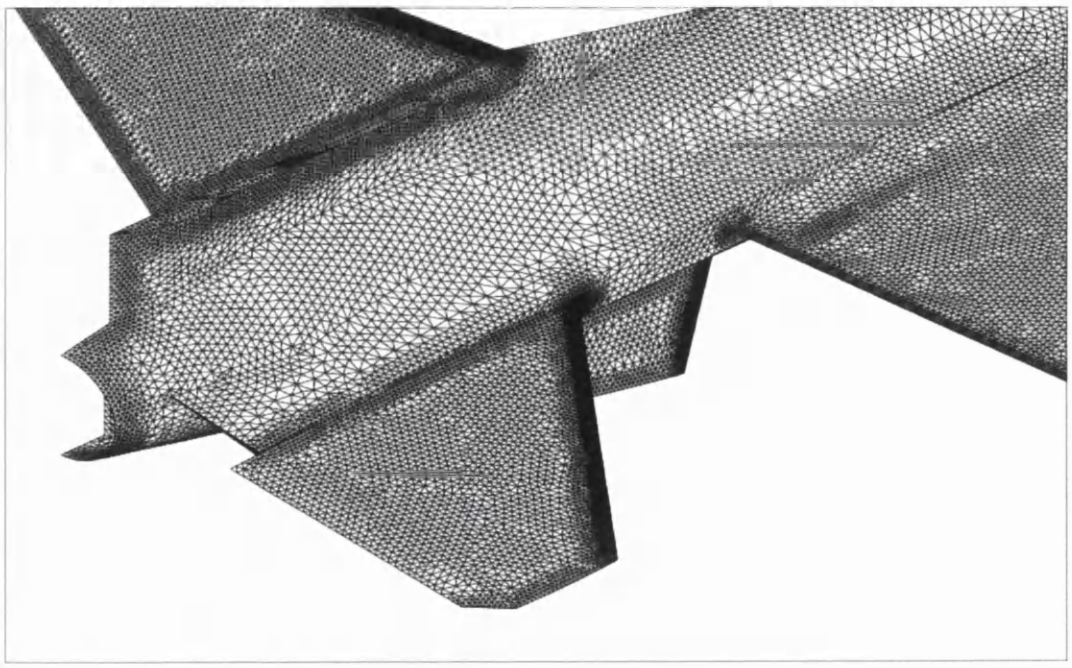
(a) Original aircraft.

(b) Super Patch mesh.

Figure 3.21 F16 fighter plane showing the original method and the Super Patch method.



(a) Original mesh

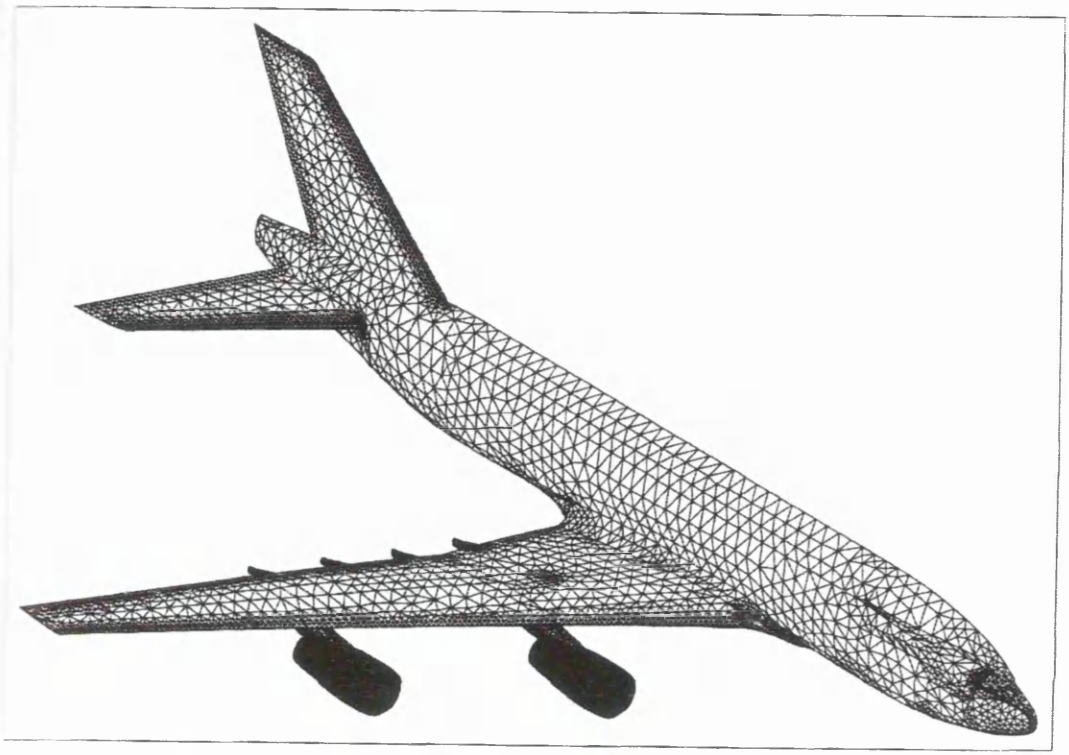


(b) Super Patch mesh.

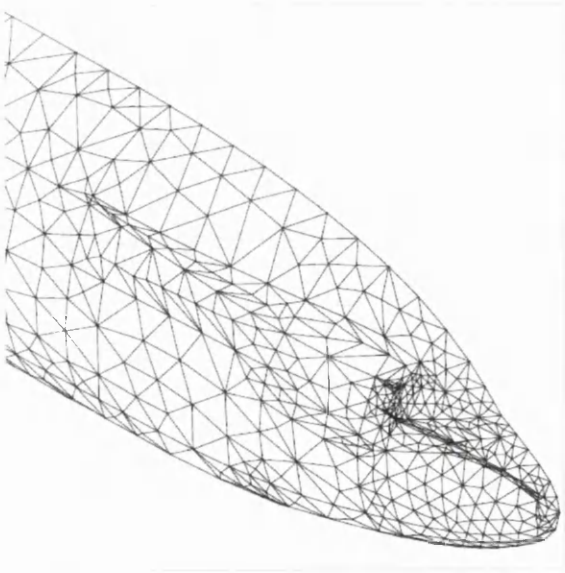
Figure 3.22 Close up of F16 fighter plane, where (a) is the original method, (b) shows the Super Patch method with no curves kept in.

The final example is of the A3XX aircraft where the cockpit of the aircraft has be designed using a number of different small and narrow patches, Figure 3.23b. In order to improve the mesh the Super Patch method has been used to eliminate these patches. Figure 3.23c shows the cockpit

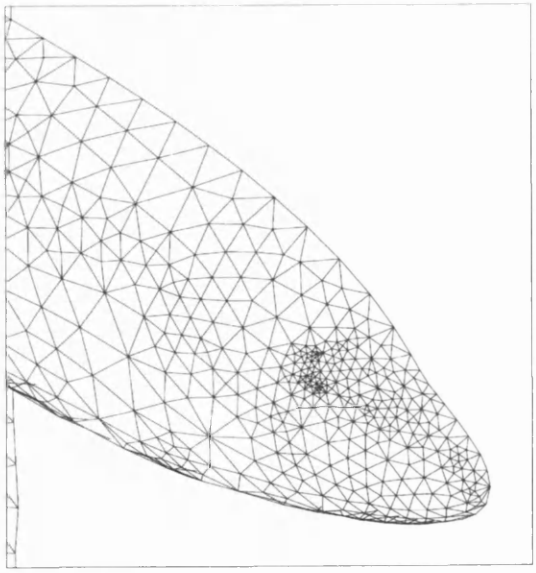
after the Super Patch method and it can be seen how the mesh has been greatly improved.



(a) Original mesh of the A3XX.

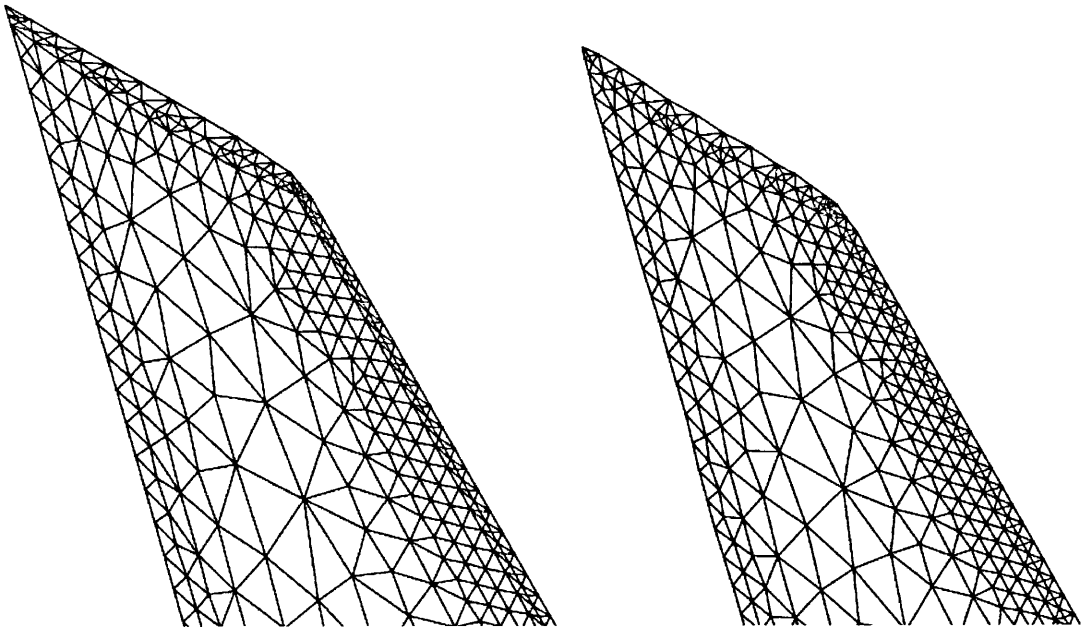


(b) Close up of the cockpit area of the original mesh.



(c) Close up of the cockpit after Super Patch method has been applied

Figure 3.25 A3XX aircraft showing a cluster of small and narrow surface patches at the front of the aircraft.



(a) Original mesh. (b) Super Patch mesh.

Figure 3.26 Close up of the tail fin.

4.

Metric Controlled Meshing

4.1 Introduction

The aim of this chapter is to introduce an alternative method to eliminate small slivers and narrow surface patches to that described in Chapter 3. The previous chapter introduced a method based on the merging together of surface patches of similar tangency, whereas this chapter will look at improving the mesh quality by the use of a metric. The role of the metric is to control the size and quality of the elements produced.

The metric controlled meshing is useful for improving poorly represented geometries and improving badly shaped elements. The process involves computing the principal curvatures and principal directions at mesh vertices. The principal curvatures at a point on a C^2 surface can be computed numerically based on the given surface triangulation [6]. This means that the surface geometry is locally approached by a quadric surface. The metric can be defined in a number of ways, the processes to be considered are:

- The elements are generated such that they comply with a specific size distribution.
- Or the element size can be defined by a metric so that the elements are appropriate to the geometry features, using curvature to determine size.

Using the given surface triangulation to build the geometry by the use of a quadric approximation (G_1) can further extend the method introduced previously. This process will be used throughout this chapter to introduce a more general approach to the problem described in Chapter 3. It should be noted that this process can also be used for geometries where the surface

definition is available. The surface geometry is used to calculate curvature and to place points onto the surface.

The improvement of the surface meshes by the method of metric re-meshing will be achieved by implementing the following processes:

- Splitting of large edges.
- Collapsing of small edges.
- Mesh smoothing.
- Swapping of mesh edges to improve mesh quality.
- Grading of mesh edges.

Algorithm

The following algorithm describes the process that the metric controlled mesh generation will take in generating the final mesh:

1 For every point:
   (a) Compute quadric surface approximation.
   (b) Compute principal curvatures.
   (c) Compute principal directions.
   (d) Calculate the metric related to every point in the tangent plane.
2 For every edge in the mesh apply the split and collapse procedures until every edge in the mesh complies with the metric.
3 Apply smoothing and swap diagonal.
4 Apply gradation to every edge in the mesh.
5 Apply smoothing and swap diagonal.

4.2 Curvature Based Meshing

This section will describe the curvature based meshing utilising a metric to control element spacing.

4.2.1 Least Squares Approximation

A frequent situation in the experimental sciences is that data collected is believed to satisfy a linear relationship of the form $y = ax + b$. However, owing to experimental error, sets of data collected at different times will rarely exhibit exactly the same relationship. Data can typically be represented

graphically as in Figure 4.1. Fitting a straight line through the data in such a way as to obtain the fit that most closely reflects the true relationship is therefore, a widespread need. One well established technique is known as the method of least squares. The approach used in this chapter is based on the method of least squares approximation of adjacent mesh vertices to compute a quadric approximation of the underlying surface.

The first process in the method is to form a local system of coordinates. For every mesh vertex *P*, the unit normal vector $\vec{v}(P)$ is calculated as a weighted average of the incident face normals as shown in Figure 4.2. The weighting is based on element size, so the smallest element will have the biggest impact on the normal. At P, the tangent plane $\Pi(P)$ is identified by the two orthogonal vectors $(\vec{\tau}_1, \vec{\tau}_2)$. The normal $\vec{\tau}_1 \wedge \vec{\tau}_2$ to the plane coincides with the normal $\vec{v}$ at P Figure 4.3 so that the local system is represented by the unit normal $\vec{v}$ and the vectors $\vec{\tau}_1$ and $\vec{\tau}_2$, with the so-called local frame $F = (P, \vec{\tau}_1, \vec{\tau}_2, \vec{v})$ at P [6, 12, 20-22, 48].

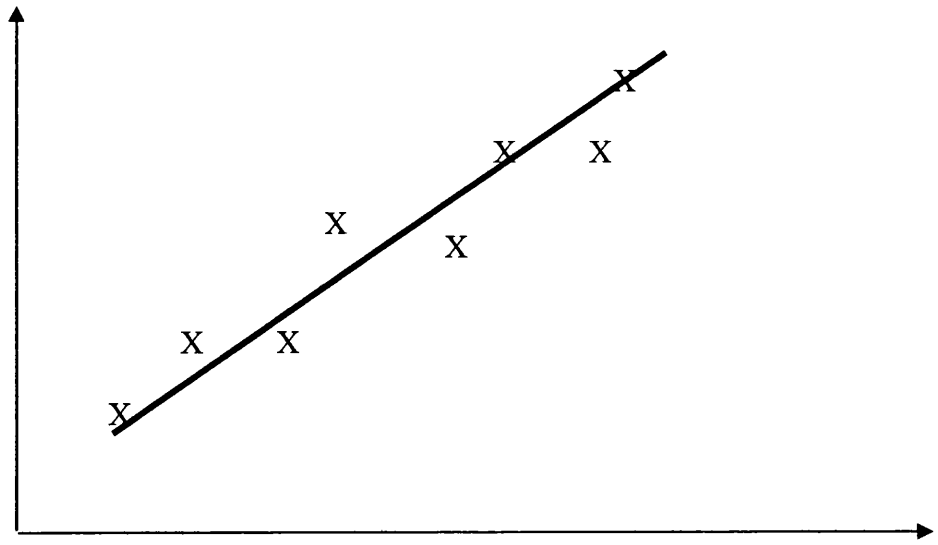


Figure 4.1 Least squares best fit.

The quadric surface $\sum$ of class $c(c \geq 2)$ centred at P is represented in the explicit form $z = z(u,v)$, where u and v represent the local directions in the tangent plane. When considering a mesh vertex P, every incident point surrounding the point must be considered in the least squares fit of a quadric

surface. For this method, consider a surface mesh point $M=(x,y,z)$, a neighbouring point to the point P=(0,0,0), where P=(0,0,0) is classed as the origin of the local coordinate system. This results in the z-axis being in the same direction of the surface normal, which leads to the equation;

$$F(x,y,z)=ax^2+bxy+cy^2-z=0 \qquad (4.2.1)$$

where a, b, and c are three coefficients to be determined.

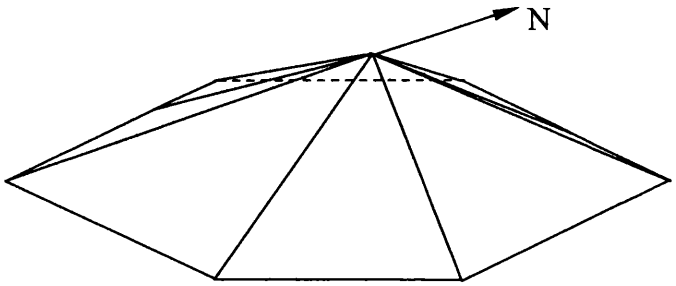


Figure 4.2 Example of weighted normal.

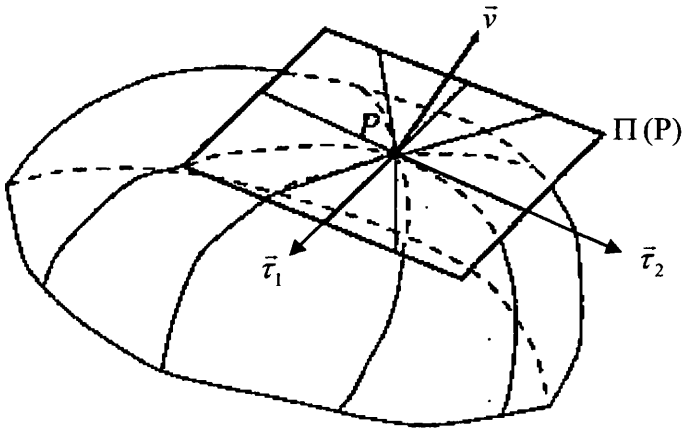


Figure 4.3 Tangent plane to the point P.

To fit the quadric surface [6], every point in the domain that is connected to the point being considered by an edge has to be looked at and assumed that the quadric surface must fit at best these points. To enable the least squares solution to take place, all points connected to the mesh vertex P must be transferred into the local system of the point under consideration. The reason for transferring points is that the least squares takes place in the local domain using the orthogonal vectors as its coordinate system. The

coordinate system for the point P is $F=(P,\vec{\tau}_1,\vec{\tau}_2,\vec{v})$, whereas the coordinate system for all other points is the Cartesian coordinate system $F=(\vec{x},\vec{y},\vec{z})$, where $\vec{x}=(1,0,0)$, $\vec{y}=(0,1,0)$ and $\vec{z}=(0,0,1)$. The transfer of coordinates requires the multiplication of each coordinate system with a 3x3 transformation matrix, where $\vec{\tau}_1$ and $\vec{\tau}_2$ are the vectors of the basis of the tangent plane

$$\overline{T}=\begin{bmatrix}\vec{\tau}_1\cdot\vec{x} & \vec{\tau}_1\cdot\vec{y} & \vec{\tau}_1\cdot\vec{z}\\ \vec{\tau}_2\cdot\vec{x} & \vec{\tau}_2\cdot\vec{y} & \vec{\tau}_2\cdot\vec{z}\\ \vec{v}\cdot\vec{x} & \vec{v}\cdot\vec{y} & \vec{v}\cdot\vec{z}\end{bmatrix} \tag{4.2.2}$$

For every point to be transferred $Q_i=\vec{s}=(x,y,z)$, the new coordinates are calculated as:

$$\vec{r}=\overline{T}\times\vec{s} \tag{4.2.3}$$

The new coordinates are stored temporarily for the calculation of the least squares fit of adjacent points. This leads to a linear system of m equations, where m represents the number of points that are connected via an edge to the point under consideration, which when solved, is the equivalent of minimizing the following sum:

$$\min\sum_{i=1}^{m}(ax_i^2+bx_iy_i+cy_i^2-z_i)^2 \tag{4.2.4}$$

This corresponds to minimizing the square of the norm of the distances to the quadric surface. In matrix form, the system can be written as:

$$\begin{pmatrix}u_1^2 & u_1v_1 & v_1^2\\ : & : & :\\ u_m^2 & u_mv_m & v_m^2\end{pmatrix}\begin{pmatrix}a\\ b\\ c\end{pmatrix}=\begin{pmatrix}d_1\\ :\\ d_m\end{pmatrix} \tag{4.2.5}$$

The difference vector d_i, is the distance from point Q_i to the tangent plane $\Pi(P)$ in the z plane. The formula for computing the least squares solution of this kind of system (AX=B), where $\overline{X}$ denotes the least squares solution of X in the equation.

$$A^tA\overline{X}=A^tB, \tag{4.2.6}$$

The resulting normal equations are then solved using a classical Gauss elimination method. Since the quadric surface is locally defined at each mesh

vertex P, it can be used to compute the local principal curvatures at each mesh vertex.

4.2.2 Principal Curvatures

The curvature is one of the shape characteristics of a surface. At every vertex the exact curvatures of the surface are usually not known explicitly for surfaces where all that is available is the surface triangulation information, therefore the radii of curvature are computed using an approximate formula. This section looks to compute the extreme values of curvature for every surface point. This is done by looking at classical differential geometry [12-15], where provided the surface is at least of order 2, the coefficients (*E, F, G* and *L, M, N)* of the first and second fundamental forms of the surface at a point P are such that:

$$\begin{aligned} \mathrm{I} &= Edu^2 + 2Fdudv + Gdv^2, \\ \Pi &= Ldu^2 + 2Mdudv + Ndv^2, \end{aligned} \tag{4.2.7}$$

The first fundamental form is used to compute the length of the curve on a surface and the second fundamental form of a surface studies the deviation of the surface from its tangent plane in the neighbourhood of the point of tangency. This leads to various developments touching on the curvature of the surface. The curvature *k* is the curvature of the normal plane determined by the directions *u* and *v* in the tangent plane.

$$k = \frac{\Pi}{\mathrm{I}} \tag{4.2.8}$$

The principal curvatures can be calculated by a direct calculation based on the above equation, as shown below:

$$k(du, dv) = \frac{\Pi}{\mathrm{I}} = \frac{Ldu^2 + 2Mdudv + Ndv^2}{Edu^2 + 2Fdudv + Gdv^2} \tag{4.2.9}$$

The variables (*E,F,G* and *L,M,N)* can be calculated so that they conform to the generality for space curves by obtaining the vector form of the equation:

$$ax^2 + bxy + cy^2 - z = 0 \tag{4.2.10}$$

This is achieved by rearranging the above equation to obtain z:

$$z = ax^2 + bxy + cy^2 \tag{4.2.11}$$

Placing this in vector form produces:

$$X = (x, y, z)$$
$$X = (u, v, au^2 + buv + cv^2) \tag{4.2.12}$$

Differentiating X with respect to u and v obtains:

$$X_u = (1,0,2au + bv)$$
$$X_v = (0,1,bu + 2cv) \tag{4.2.13}$$

The variables can be calculated by using the dot product as:

$$E = X_u \cdot X_u, \quad F = X_u \cdot X_v, \quad G = X_v \cdot X_v,$$

$$E = (1 + 2au + bv) \cdot (1 + 2au + bv)$$
$$E = 1 + (2au + bv)^2$$
$$F = (1 + 0 + 2au + bv) \cdot (0 + 1 + bu + 2cv)$$
$$F = (2au + bv)(bu + 2cv)$$
$$G = (0 + 1 + bu + 2cv) \cdot (0 + 1 + bu + 2cv)$$
$$G = 1 + (bu + 2cv)^2$$

The variables *L, M, N* are computed using the second differential of the vector and the normal, where the normal is computed as;

$$\vec{\nu} = \frac{X_u \times X_v}{|X_u \times X_v|} = (0,0,1) \tag{4.2.14}$$

and the coefficients are calculated as:

$$L = X_{uu} \cdot \vec{\nu}, \quad M = X_{uv} \cdot \vec{\nu}, \quad N = X_{vv} \cdot \vec{\nu}.$$

$$L = 2a,$$
$$M = b, \tag{4.2.15}$$
$$N = 2c.$$

In this case the local frame is considered where P=(0,0), therefore,

$$E = 1, \qquad F = 0, \qquad G = 1,$$
$$L = 2a, \qquad M = b, \qquad N = 2c.$$

The analysis of the variations of the normal curvature function leads to resolve the following equation:

$$\det \begin{vmatrix} du^2 & -dudv & dv^2 \\ E & F & G \\ L & M & N \end{vmatrix} = 0. \tag{4.2.16}$$

This equation in principle admits two distinct solutions - 2 pairs (λ_1, κ_1), (λ_2, κ_2). The extreme values κ_1 and κ_2 of κ_n (i.e., the roots of the equation)

are the so-called principal curvatures of the surface at P. Comparing equation 4.2.16 with the second order equation;

$$\kappa^2 - (\kappa_1 + \kappa_2)\kappa + \kappa_1\kappa_2 = 0 \tag{4.2.17}$$

Yields;

$$\begin{aligned} \kappa_1\kappa_2 &= \frac{LN - M^2}{EG - F^2} = 4ac - b^2, \\ \kappa_1 + \kappa_2 &= \frac{NE - 2MF + LG}{EG - F^2} = 2(a + c), \end{aligned} \tag{4.2.18}$$

The Gaussian curvature $K = \kappa_1\kappa_2$ and the mean curvature of the surface at P $H = \frac{1}{2}(\kappa_1 + \kappa_2)$ are derived from the equation above. Replacing *E, F, G, L, M, N* from equation 4.2.15 allows the extreme values κ_1 and κ_2 at P to be found;

$$\kappa_i = \frac{2(a + c) \pm \sqrt{\Delta}}{2} \tag{4.2.19}$$

with $\Delta = (2(a + c))^2 - 4(4ac - b^2)$.

4.2.3 Principal Directions

To compute the principal directions, the principal curvatures will be used. This means that the principal directions lie along the path of the principal curvatures. From equation 4.2.16 it can be seen that two solutions define two directions in the (u, v) plane. Taking the determinant of the matrix in equation 4.2.16 results in;

$$dv^2 b + dudv(2c - 2a) - du^2 b = 0, \tag{4.2.20}$$

The corresponding directions in the tangent plane are the principal directions. Hence, solving equation 4.2.16 leads to the root:

$$\lambda_{1,2} = \frac{2(c - a) \pm \sqrt{4((a - c)^2 + b^2)}}{2b}, \tag{4.2.21}$$

To find the expressions of the two vectors, employ Euler's theorem and consider the relation:

$$\lambda_{1,2} = \frac{du}{dv} = \tan(\theta), \tag{4.2.22}$$

From the above equation the angle θ can be calculated so that the orthogonal vectors can be aligned with the principal curvatures. The local frame leads to the two vectors in the global frame, namely

$$\begin{aligned}\vec{\lambda}_1 &= e_{1x,}.\vec{\tau}_1 + e_{1y,}.\vec{\tau}_2, \\ \vec{\lambda}_2 &= e_{2x,}.\vec{\tau}_1 + e_{2y,}.\vec{\tau}_2,\end{aligned} \tag{4.2.23}$$

with,

$$\vec{e}_1 = (\cos(\theta), \sin(\theta))^t, \vec{e}_2 = (-\sin(\theta), \cos(\theta))^t.$$

The principal directions make up a 3x3 matrix including the orthogonal vectors that have been rotated into line with the principal curvatures and the normal of the point. This is represented as;

$$D(P) = \begin{bmatrix} \vec{\lambda}_1 \\ \vec{\lambda}_2 \\ \vec{v} \end{bmatrix} \tag{4.2.24}$$

4.2.4 Geometric Metric G_3

For surface meshes, the validity of the geometric approximation within a given tolerance is related to the control of the distance between the mesh elements and the underlying surface. Hence, the mesh is such that all vertices belong to the surface, all elements are close to the surface and any element is close to the tangent plane at its surface vertices. The first two requirements enable the distance (gap) between the mesh and the surface to be bounded. The control of the gap between an edge and the surface is performed by using a metric G_3, which still needs to be defined. Choosing an adequate G_3 means that the edge will not be further from the surface than a given threshold value [6, 48, 16]. The length of an edge AB is computed by using the metric;

$$l_{AB} = \left\| \overrightarrow{AB} \right\|_{G3} = \sqrt{\overrightarrow{AB}^t G_3 \overrightarrow{AB}} \tag{4.2.25}$$

The edge conforms to the metric if;

$$l_{AB} = 1 \quad \text{for } G_3 \tag{4.2.26}$$

A choice of metrics is available depending on what mesh is to be generated. A metric of the form shown below will specify a mesh of uniform sizes *h* on the surface.

$$G_3 = \begin{pmatrix} \frac{1}{h^2} & 0 & 0 \\ 0 & \frac{1}{h^2} & 0 \\ 0 & 0 & \frac{1}{h^2} \end{pmatrix} \tag{4.2.27}$$

A metric of the form shown below will specify a variable field of sizes on the surface, such that *h* depends on the position.

$$G_3 = \begin{pmatrix} \frac{1}{h^2(P)} & 0 & 0 \\ 0 & \frac{1}{h^2(P)} & 0 \\ 0 & 0 & \frac{1}{h^2(P)} \end{pmatrix} \tag{4.2.28}$$

An isotropic control of the mesh in relation to the radii of curvature can be obtained, when $h(P) = \alpha\rho(P)$, where $\rho(P)$ is the smallest of the radii of curvature ρ_1 and ρ_2 at the point P and α is an adequate coefficient.

A metric of the form shown below may be used to control the mesh generation based on curvature.

$$G_3(P)_{\rho_1,\rho_2} = D^t(P) \begin{pmatrix} \frac{1}{\alpha^2\rho_1^2(P)} & 0 & 0 \\ 0 & \frac{1}{\beta^2\rho_2^2(P)} & 0 \\ 0 & 0 & \lambda \end{pmatrix} D(P), \tag{4.2.29}$$

The metric denoted as $G_3(P)_{\rho 1,\rho 2}$ is constructed in the tangent planes of the mesh vertices of the given domain. *D(P)* corresponds to the principal directions at P, $\rho_1 = 1/\kappa_1$, $\rho_2 = 1/\kappa_2$ are the main radii of curvature, where κ_1 and κ_2 are as computed earlier. α and β are appropriate coefficients and $\lambda \in \Re$, provides an anisotropic curvature based control of the geometry. It can be seen that the metric prescribes mesh sizes and stretching directions at mesh vertices, which means that the local element size is proportional to the

principal radii of curvature and the coefficient of proportionality is related to the largest allowable deviation gap between the mesh elements and the surface geometry [6].

Consider a metric of the form given in equation 4.2.28 and choose $h(P) = \alpha\rho(P)$. The aim is to fix α so as to obtain control within a given ε of the gap between the mesh and the surface, by considering the osculating circle where the curve having a second-order contact with the planar curve $\gamma(s)$ at P, which is denoted as ψ. The centre C_p of ψ is related to the radius of curvature ρ by;

$$C_p = P + \rho\vec{v} \tag{4.2.30}$$

where $\vec{v}$ is the main unit normal vector to $\gamma(s)$ at P. Equation 4.2.30 is an approximation of the order two at the curve intersection of the surface with any plane supported by $\vec{v}$ with ρ the radius of curvature in this particular plane, Figure 4.4

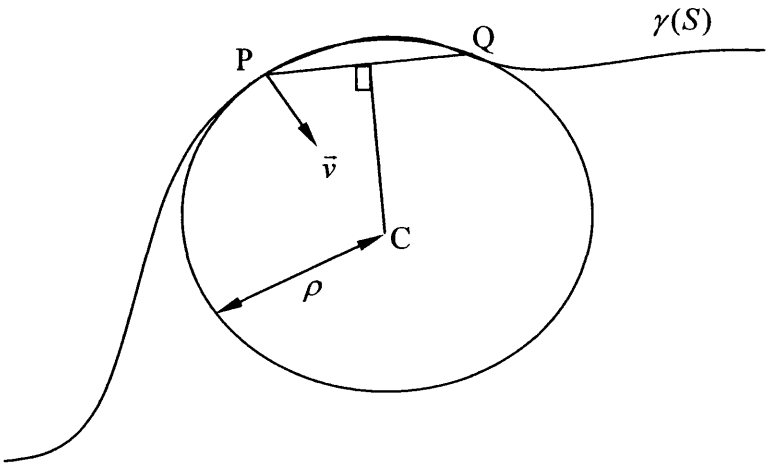


Figure 4.4. Osculating circle approximation.

Approximating the surface with an edge while maintaining control of the accuracy is equivalent to controlling the gap between a discretisation of the osculating circle of the plane.

It can be seen that fixing $\alpha = 1$ is equivalent to discretising this circle with 6 edges. The length of the circle for the metric given in equation 4.2.28 is $2\pi \approx 6$. Then, as shown in Figure 4.5a taking an edge of length ρ, the

relationship, $\rho - \delta = \sqrt{\rho^2 - (\alpha\rho/2)^2}$ is obtained. By rearranging and placing $\alpha = 1$, $\delta = \rho(1 - \sqrt{3/4})$ and thus $\delta/\rho = 1 - \sqrt{3/4} \approx 0.15$. The relative gap to the circle is 15%.

On the other hand, if it is assumed that the discretized edge is of size $\alpha\rho$, then the following equation is obtained;

$$\delta = \rho(1 - \sqrt{1 - \frac{\alpha^2}{4}}) \tag{4.2.31}$$

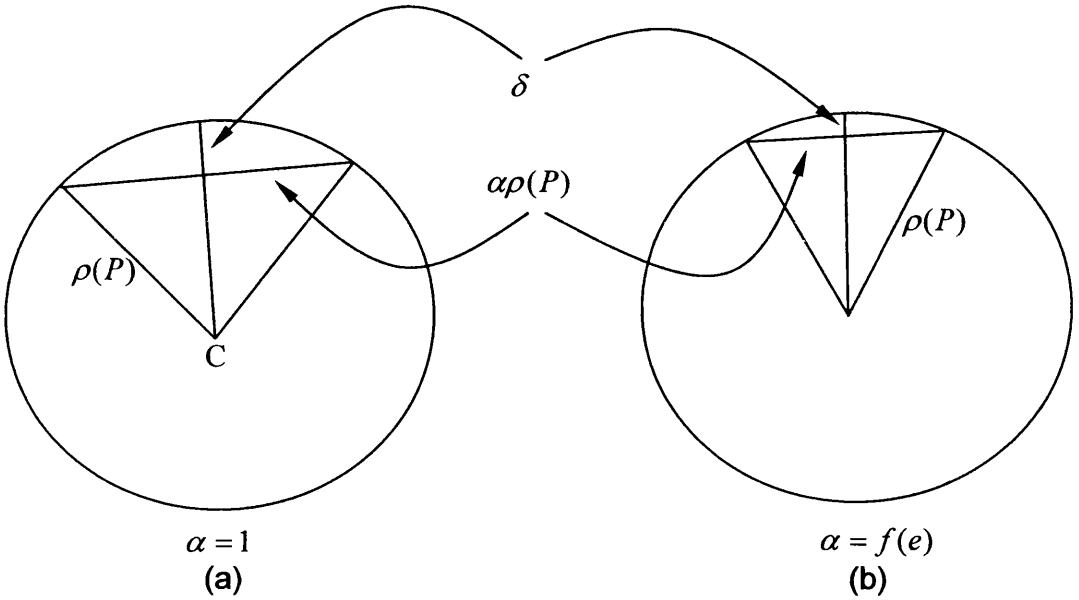


Figure 4.5: Discretisation of a circle with step size $\rho(P)$ (a), or with a step size $\alpha\rho(P)$ (b).

Thus setting $\delta/\rho \leq \varepsilon$ comes down to fixing:

$$\alpha \leq 2\sqrt{\varepsilon(2-\varepsilon)}, \tag{4.2.32}$$

For the anisotropic case, the control metric is that of equation (4.2.29) in which the coefficients α and β are involved. Since the deviation from the circle is not the same in the two principal directions and thus the value varies according to the direction, setting a constant gap comes down to setting δ in both directions to be equal and fixing

$$\alpha \leq 2\sqrt{\varepsilon(2-\varepsilon)}, \tag{4.2.33}$$

and defining

$$\beta = 2\sqrt{\varepsilon\frac{\rho_1}{\rho_2}(2-\varepsilon\frac{\rho_1}{\rho_2})} \tag{4.2.34}$$

means that we have a local absolute control, within ε, of the distance between an edge and the surface, depending on the directions.

In finite element computations it may be important to allow an isotropic based curvature at each mesh vertices. The reason for this is to achieve a more acceptable computational mesh.

Figure 4.6 shows the effect of the variable ε on the final mesh. In each picture only the variable ε has been changed. Figure 4.6a shows the original mesh with 226 vertices and 448 elements. For this example five values of ε have been used, $\varepsilon = 0.01$ Figure 4.6b, $\varepsilon = 0.005$ Figure 4.6c, $\varepsilon = 0.001$ Figure 4.6d, $\varepsilon = 0.0005$ Figure 4.6e, and finally $\varepsilon = 0.0001$ Figure 4.6f which produced a mesh with 20316 vertices and 40628 elements. It can be seen that the smaller the value of ε the more vertices are produced. The reason for this is that the gap that is allowed between the edge and the surface is reduced, so more elements are needed to represent the final mesh to comply with the given tolerance.

4.3 Split and Collapse Algorithms

The metric controls the size of an edge at any given vertex A within the domain as defined in section 4.2.5. In order to achieve the specified size, the edge being considered has to be either split or collapsed. This section will introduce these two methods.

In order to decide if an edge complies to the given size specification, the length of the segment $\overrightarrow{AB}$ in the local domain has to be computed as l_{AB} and since the metric is independent of the position it makes it possible to reduce the problem to the Euclidean case [6] as shown below;

$$l_{AB} = \left\|\overrightarrow{AB}\right\|_{G3} = \sqrt{\overrightarrow{AB}^t G_3 \overrightarrow{AB}}\,. \tag{4.3.1}$$

The metric is used to control the re-meshing of the geometry and a mesh vertex is said to conform to the metric if:

$$1/\sqrt{2} \le l_{AB} \le \sqrt{2} \tag{4.3.2}$$

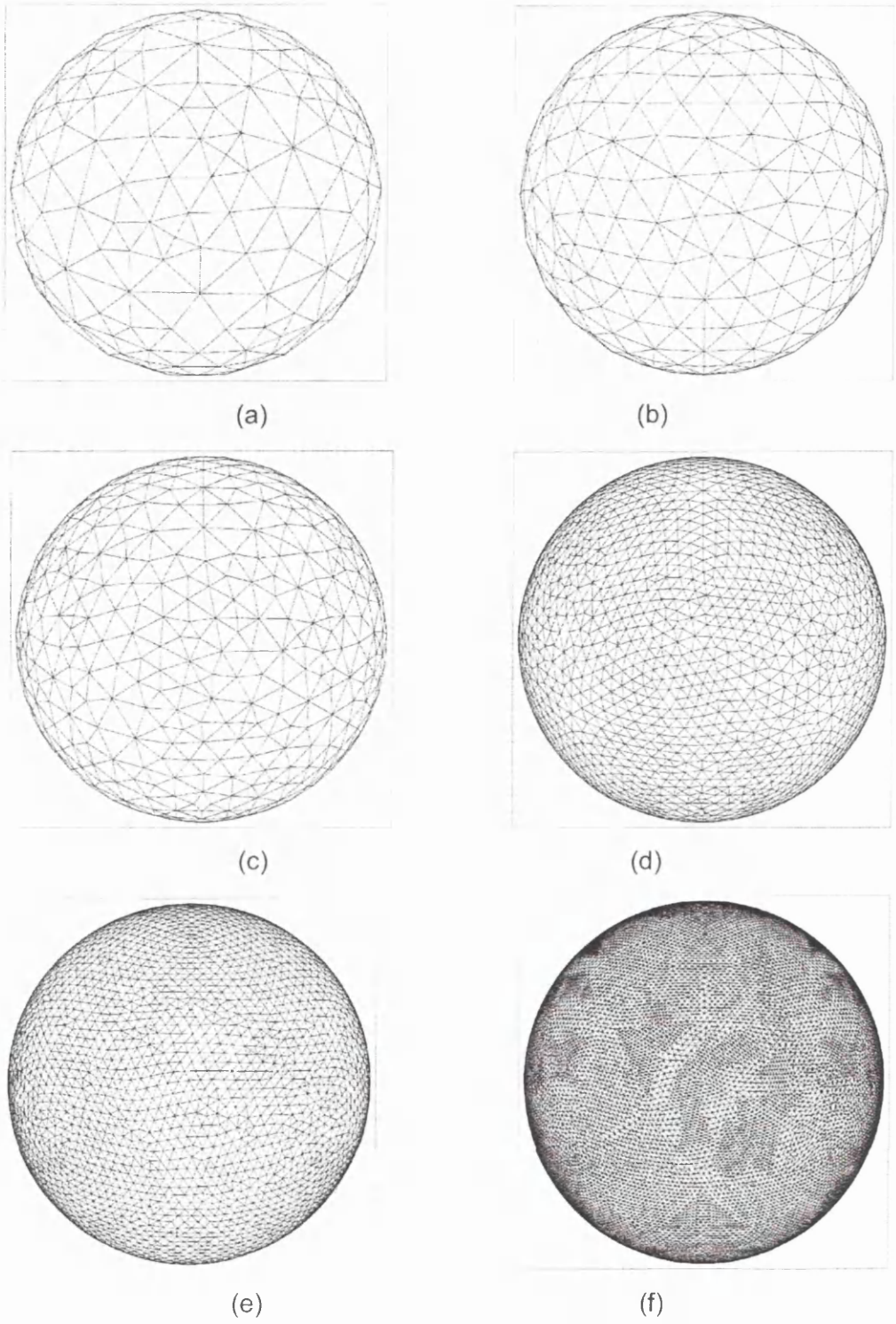


Figure 4.6 Effect of variation of ε.

If the length $l_{AB} > \sqrt{2}$ for any mesh vertex within the domain then the edge being considered is split. If the length of any mesh edge is computed as $l_{AB} < 1/\sqrt{2}$ then the edge is collapsed. These processes will be continued

until the final mesh complies with the metric and the mesh is expressed as a unit mesh.

The optimization procedure of splitting long edges and collapsing short edges is in order to produce a mesh that complies to any given metric. The split and collapse procedures are described below:

- *Split process;* this is used to enrich the mesh in areas of high curvature. The edge being considered is split if $l_{AB} > \sqrt{2}$, so a node is placed along the mid-point of the edge using the G_1 approximation. The elements are split by replacing the two triangles sharing the edge by four new ones (Figure 4.7).

Collapse process; this is used to increase the length of edges in areas of lower curvature. The edge being considered is collapsed if $l_{AB} < 1/\sqrt{2}$, the edge is removed and the two nodes making the edge are replaced by one node at the centre of the edge, which is calculated using the G_1 approximation. For edges that are connected to a ridge node and an internal node, the edge is collapsed to the ridge point in order to keep a true representation of the geometry. The same applies for any node that is connected to a corner node, in this case the edge is collapsed to the corner node. The collapse procedure for all internal nodes can be seen in Figure 4.8 (a) where the edge is collapsed to a node placed at the centre of the edge. The special case examples can be seen in Figures 4.8 (b) and (c).

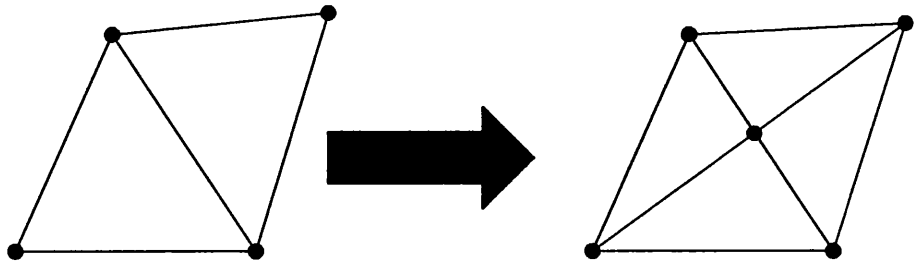


Figure 4.7 Mesh enrichment.

4.4 G1 Approximation

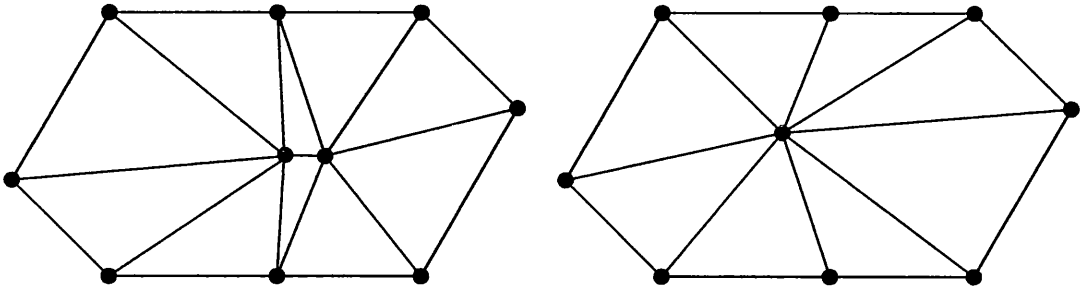
Throughout the mesh generation process the method used continuously looks to place points onto the geometrical surface. If the surface definition is available then this process becomes simpler and the points generated can be placed using the mapping process as described in Chapter 2. In cases where the only information available is from the surface triangulation, then a different method has to be employed. Problems of this type were first looked at by Nielson [7] in 1977 where he looked at interpolating the triangle to obtain the position of the new point. These ideas were later considered by Weatherill [11] where he employed the method for mesh adaptation techniques. The method G_1 reconstructs the surface geometry using a transfinite, visually continuous, triangular interpolant. The interpolate utilises the outward normals, unlike such methods as the Ferguson patch which uses partial derivatives on boundaries.

The G_1 representation of the surface has a continuously varying outward normal vector. The method can be described using a surface segment which is a vector-valued function represented by

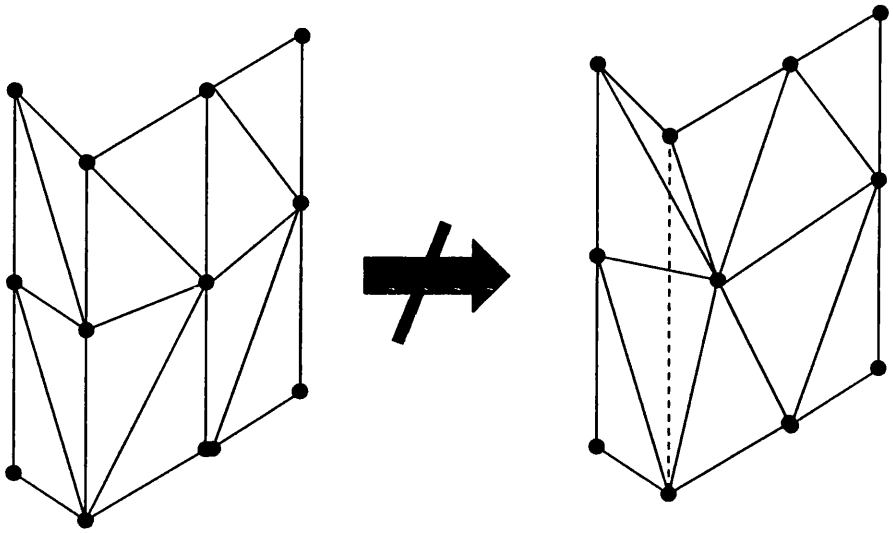
$$F(b_1,b_2,b_3)=\begin{cases} x(b_1,b_2,b_3) \\ y(b_1,b_2,b_3) \\ z(b_1,b_2,b_3) \end{cases} \tag{4.4.1}$$

with $(b_1,b_2,b_3)\in T$, where the domain T is defined as

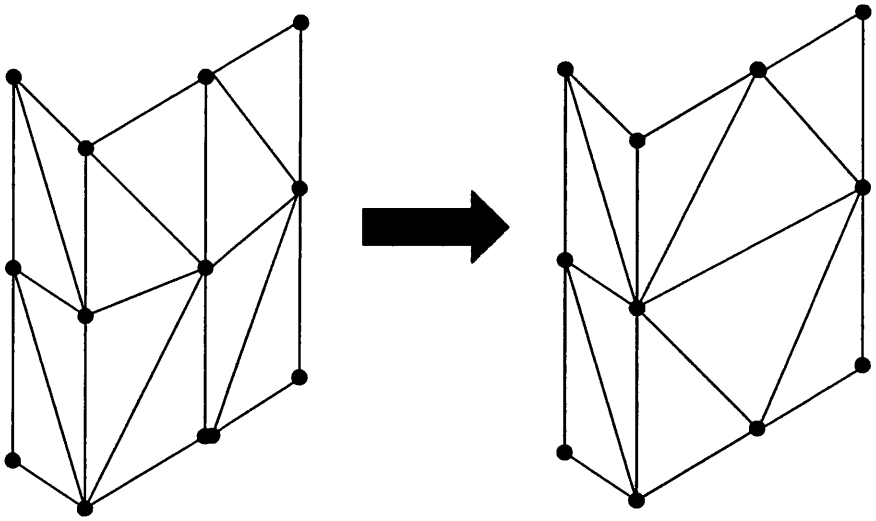
$$T=\{(b_1,b_2,b_3): 0\le b_i \le 1, i=1,2,3; b_1+b_2+b_3=1\} \tag{4.4.2}$$



(a) Shows collapse for internal vertex.



(b) Collapse algorithm of a ridge point using the same procedure as for internal nodes, it shows how the geometry definition is lost.



(c) Collapse procedure of a ridge point showing how the geometry definition is kept by collapsing to another ridge point.

Figure 4.8 Collapse procedures for edges.

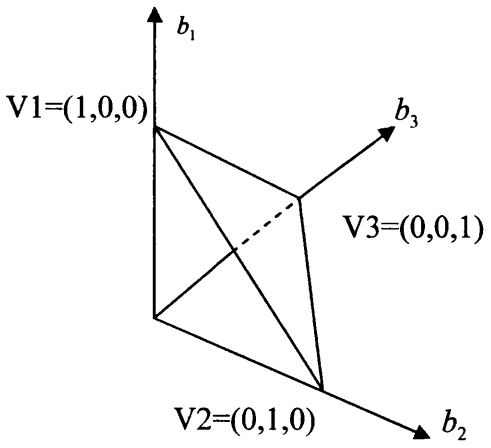


Figure 4.9 Schematic of the surface triangle.

These are known as area coordinates, which control the weighting along each edge of the segment. If a node is to be placed at the centre of the segment then the weight will be $b_1 = b_2 = b_3 = 1/3$ in all cases. For the case where the new point will be placed along the edge then the weight will be:

Along edge e1: $b_1 = 0.0 \quad b_2 = 0.5 \quad b_3 = 0.5$

Along edge e2: $b_1 = 0.5 \quad b_2 = 0.0 \quad b_3 = 0.5$

Along edge e3: $b_1 = 0.5 \quad b_2 = 0.5 \quad b_3 = 0.0$

The edge notation can be seen in equation 4.4.2.

The canonical form for the triangle can then be set up where the vertices of T are denoted by:

$$V_1 = (1,0,0), \qquad V_2 = (0,1,0), \qquad V_3 = (0,0,1).$$

Figure 4.9 shows such a triangle in schematic form. The edges are defined by;

$$e_1 = \{(0,b_2,b_3) \in T\}, \qquad e_2 = \{(b_1,0,b_3) \in T\}, \qquad e_3 = \{(b_1,b_2,0) \in T\}.$$

This is shown in Figure 4.10.

The univariate cubic Hermite operator along edges (Figure 4.11) produces a cubic interpolant from end points V_0 and V_1 and the tangent vectors V_0' and V_1' can be written as

$$H[V_0,V_1,V_0',V_1'](t) = h_0(t)V_0 + h_1(t)V_1 + H_0(t)V_0' + H_1(t)V_1', \qquad (4.4.3)$$

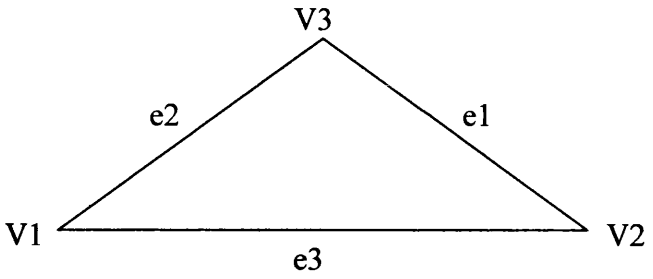


Figure 4.10 Edge vertex notation

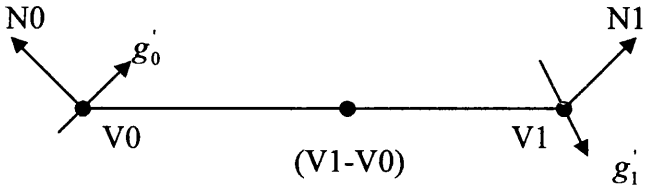


Figure 4.11 Schematic of the edge of a triangular patch

The blending functions used are,

$$h_0(t) = 1 - 3t^2 + 2t^3, \qquad h_1(t) = 3t^2 - 2t^3, \tag{4.4.4}$$

$$H_0(t) = t - 2t^2 + t^3, \qquad H_1(t) = t^3 - t^2.$$

$0 \le t \le 1$, and a prime denotes differentiation with respect to the parametric variable t.

The normals at the points are N_0 and N_1 the tangent vectors which are unknown in our procedure are V_0' and V_1'. The tangent at each point is computed using the normal at each point, making sure that V_0' be in the plane containing N_0 and $V_1 - V_0$, and that V_1' be in the plane of N_1 and $V_1 - V_0$.

Given this data, the transfinite interpolant can be defined. The interpolant used is based upon the ideas of the side vertex method.

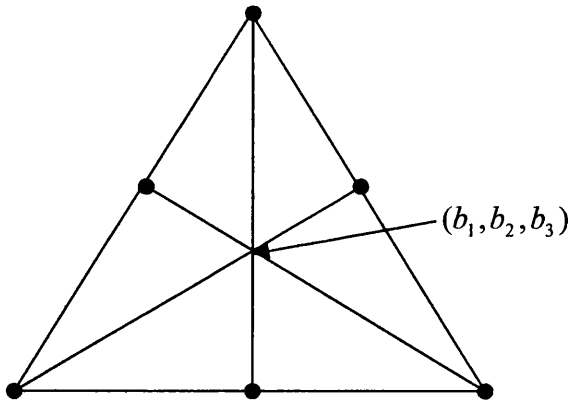


Figure 4.12 Interpolant.

The arguments of the interpolant are a vertex and a point on the opposite facing edge, where a ray emanating from the vertex and passing through the point (b_1, b_2, b_3) intersects (Figure 4.12). The boundary points can be expressed as;

$$c = \frac{b_j V_j + b_k + V_k}{1 - b_i}. \tag{4.4.5}$$

The interpolant is given by,

$$G_i[F](b_1, b_2, b_3) = g[F(V_i), F(c), N[F](V_i), N[F](c)](1 - b_i), \tag{4.4.6}$$

where $N[F](V_i)$ is the outward surface normal vector at the vertex V_i, which leads to:-

$$G[F] = W_1 G_1[F] + W_2 G_2[F] + W_3 G_3[F] \tag{4.4.7}$$

where:-

$$W_i = \frac{b_j^2 b_k^2}{b_1^2 b_2^2 + b_2^2 b_3^2 + b_3^2 b_1^2}. \tag{4.4.8}$$

are the weighting values at each edge. This can be written as;

$$W_i = \frac{b_j b_k}{b_1 b_2 + b_2 b_3 + b_3 b_1}. \tag{4.4.9}$$

Nielson [7] proved that (4.4.6)-(4.4.8) satisfy the interpolation conditions

$$G(F)(\delta T) = F(\delta T), \qquad N[G(F)](\delta T) = N[F](\delta T), \tag{4.4.10}$$

where δT refers to the boundaries of the triangle t. The interpolation defined by equation 4.4.7 represents a transfinite continuous patch. The function values along an edge are taken as cubic Hermite interpolants and the cross

boundary normal derivatives as linear interpolants. This means that the entire triangular approximation depends only on the nine values at the vertices. This section will look at an example to obtain a six-parameter, G_1 interpolant. This will depend solely on the position and outward surface normal at the three vertices of T. Given the transfinite formulation, the discretised form for each of the three radial projectors is given by:-

$$g_i(b_1,b_2,b_3) = g\left\{F_i, g[F_j,F_k,N_j,N_k](\frac{b_jV_j+b_kV_k}{1-b_i}) \bullet N_i, N_j(\frac{b_jV_j+b_kV_k}{1-b_i})\right\}(1-b_i).$$

The interpolant involves three vertex-to-side interpolations. This approach ensures that the added points are consistent with a G_1 representation of the surface configuration.

The accuracy of the technique has been examined by Weatherill [11] for unstructured meshing of the surface of a sphere with unit radius. Figure 4.13 clearly shows the benefit of using the G_1 approximation, since the sphere generated using linear interpolation produces peaks in the mesh. The G_1 mesh does not produce these peaks and the accuracy of the final mesh has greatly improved.

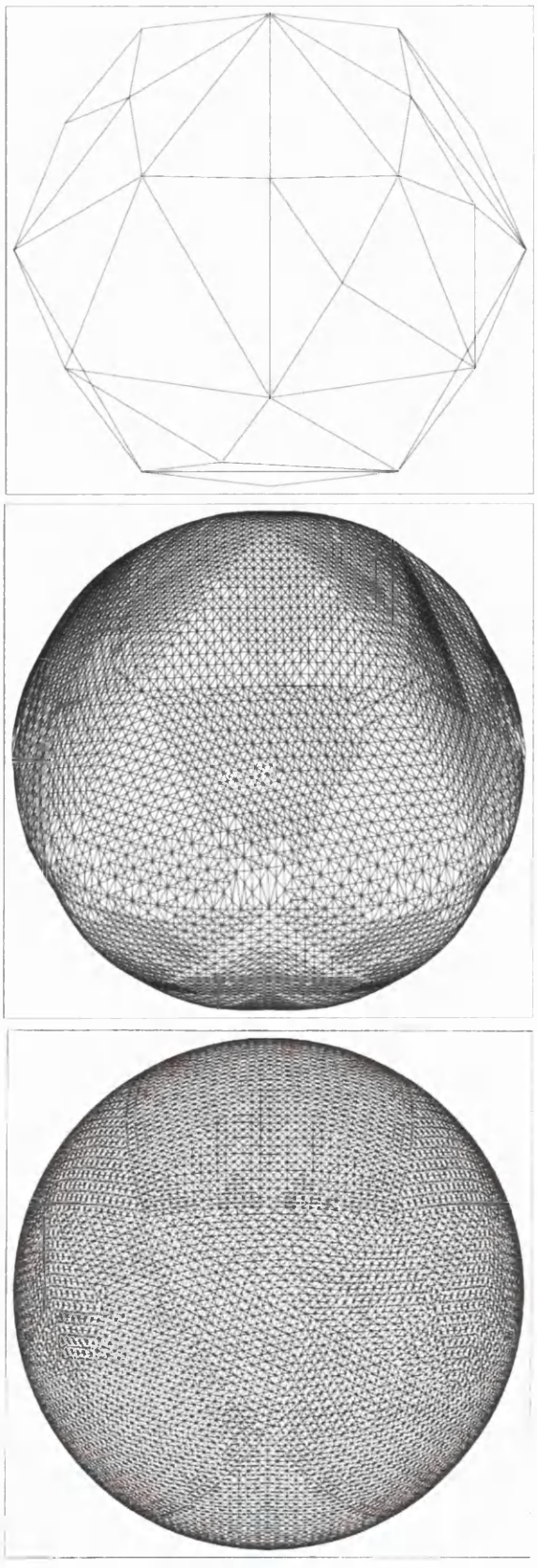


Figure 4.13 Surface reconstruction: (a) initial surface triangulation; (b) surface triangulation using linear interpolation for point position; (c) surface triangulation using G_1 interpolation for point position.

4.5 Dealing with Ridge and Corner Points

Ridge and corner points have to be preserved in the final mesh to enable an accurate representation of the geometry to be retained. The definition of a ridge edge is when the dihedral angle μ between two faces exceeds a prescribed value (see Figure 4.14) with a default value of 60 degrees employed in this work. Corners are defined when three incident ridges meet at a vertex or if two ridges form a small angle (Figure 4.15).

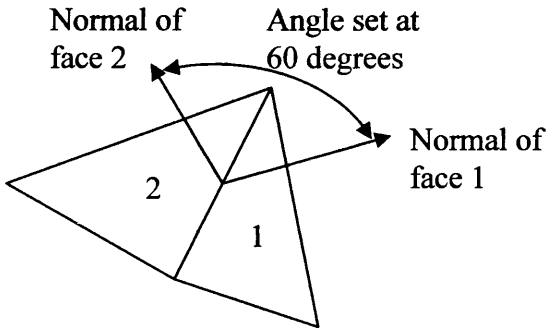


Figure 4.14 Angle between faces to describe ridge edges.

In cases where the surface definition is available, the user can flag which curves in the make up of the geometry are ridges in a data file and hence the ridge information is read automatically. From this information the corner points can be computed as previously described. The benefits of this process over the previous method are that the user is defining the ridge curves and hence they have control over what curves are kept. In the previous method the facet information is used to compute ridge edges, which can cause ridges to be produced in areas where they are not wanted.

An example of this can be seen in Figure 4.16, where Figure 4.16a shows the m6 wing where the ridges were computed using the facet information, whereas, Figure 4.16b highlights the ridges flagged manually. This example showed that using the facet definition produced ridges along the leading edge of the wing where they are not wanted, whereas the data file has eliminated this problem, as the user has defined which curves are ridges.

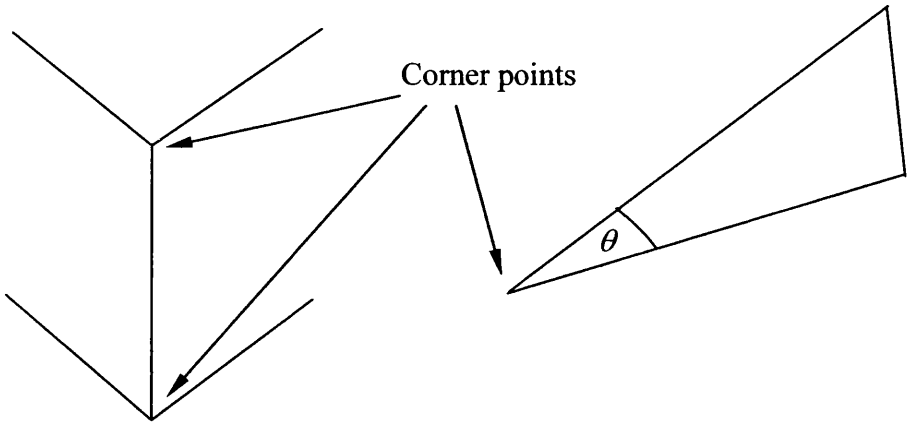
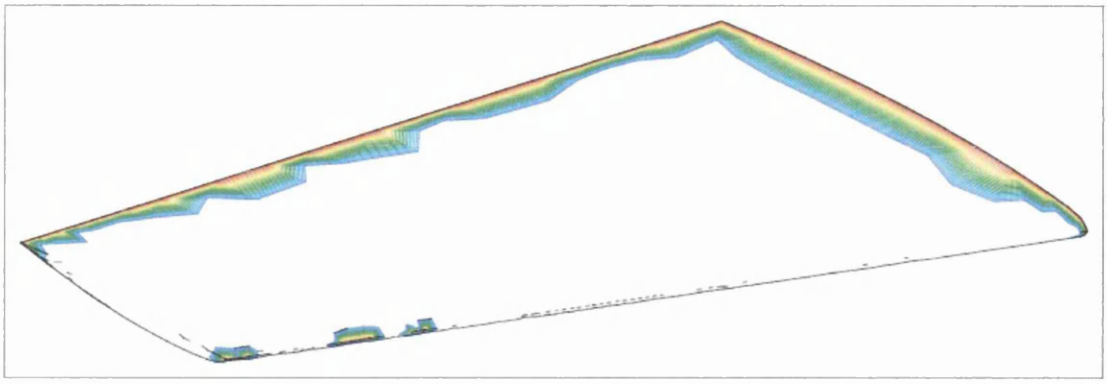


Figure 4.15 Example of corner points.

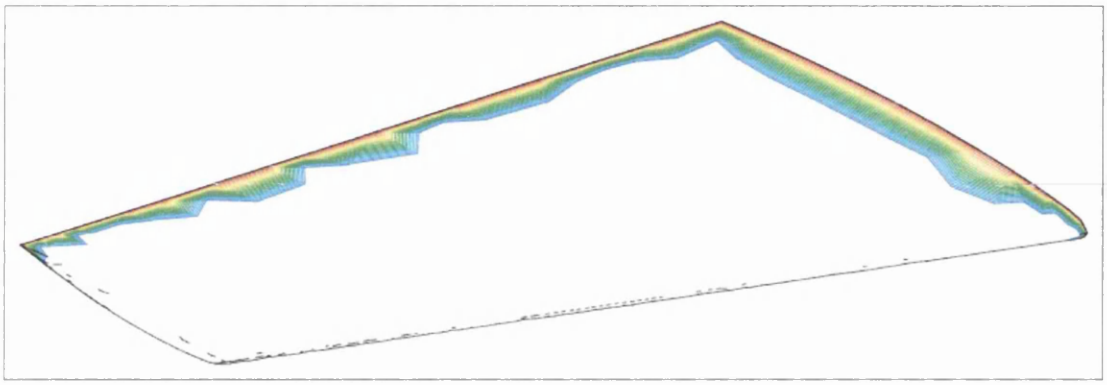
The effect of this on the resulting mesh can be seen in Figure 4.17 where (a) shows the mesh produced using the facet information to identify ridges. Comparing Figure 4.16(a) and Figure 4.17(a) it can be seen how the areas highlighted have affected the resulting mesh. Figure 4.17(b) shows the resulting mesh produced utilising the data file. It can be seen that the mesh along the leading edge of the wing has improved producing a more acceptable mesh for finite element calculations.

The reason for this problem is due to the original surface mesh not being an accurate representation of the surface geometry. This means that the facet method for calculating ridges will produce this result, whereas if the original surface mesh could be a more accurate representation then this problem would not occur.

Because the G_1 approximation is used for placing points onto the surface, care has to be taken when placing a vertex along a ridge edge so that the point is placed in the correct position. By computing the normal on either side of the ridge, the normal related to the element being considered during the G_1 calculation Figure 4.18 can be utilised. This is done to preserve a true representation of the under lying surface geometry.

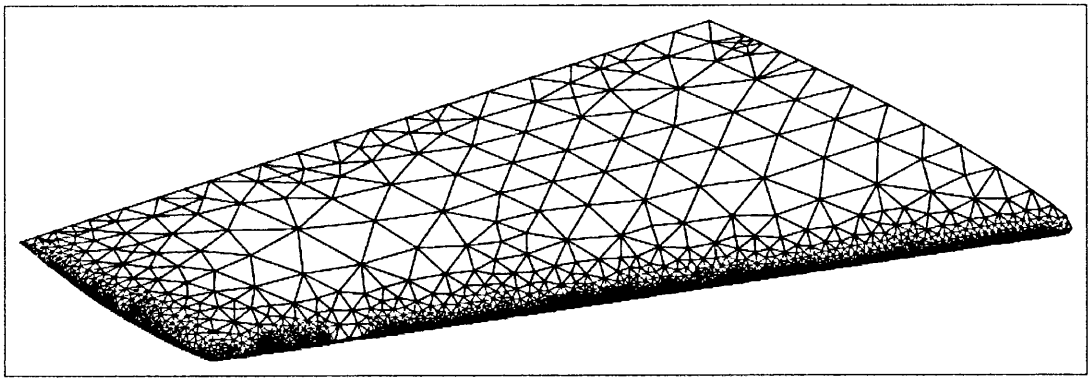


(a) Ridges as computed from the facet information.

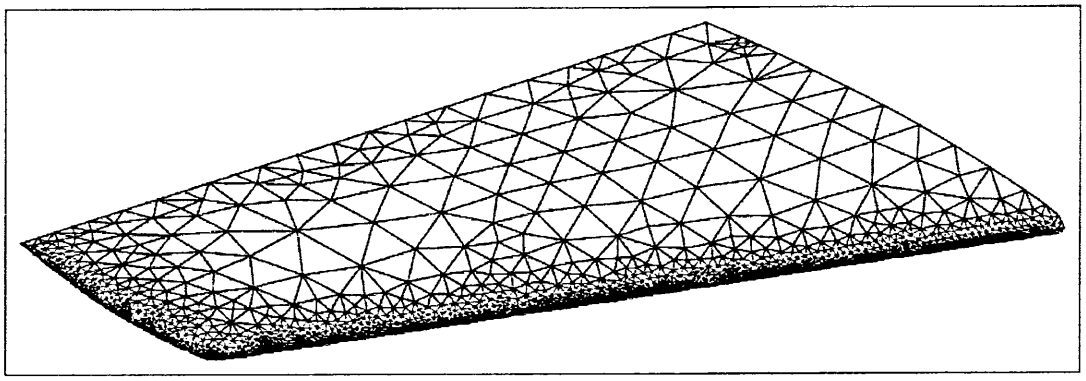


(b) Ridges as computed using the data file.

Figure 4.16 Example of ridge calculation, the highlighted areas identify ridges.



(a) m6 wing, ridges computed using facet information.



(b) m6 wing, ridge computed using data file.

Figure 4.17 m6 wing, effect of ridge calculation.

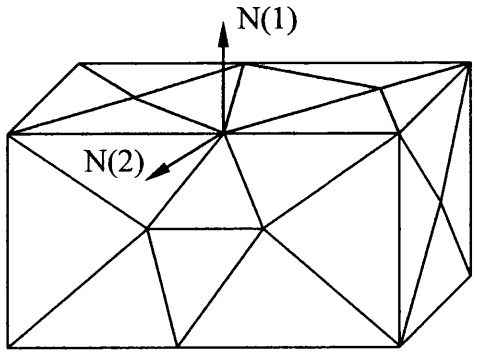


Figure 4.18 Two normals are computed on a ridge vertex.

4.6 Mesh Gradation

The metric produced in section 4.2.5, may cause the element size to change rapidly from one vertex to another, the mesh gradation may not be bounded locally, Figure 4.19; thus leading to poorly-shaped elements. To overcome this problem the mesh is graded utilising an edge based method [48].

The gradation procedure will be applied after the mesh has been generated to comply with the metric G_3. The gradation procedure will ensure a mesh of high quality for finite element calculations.

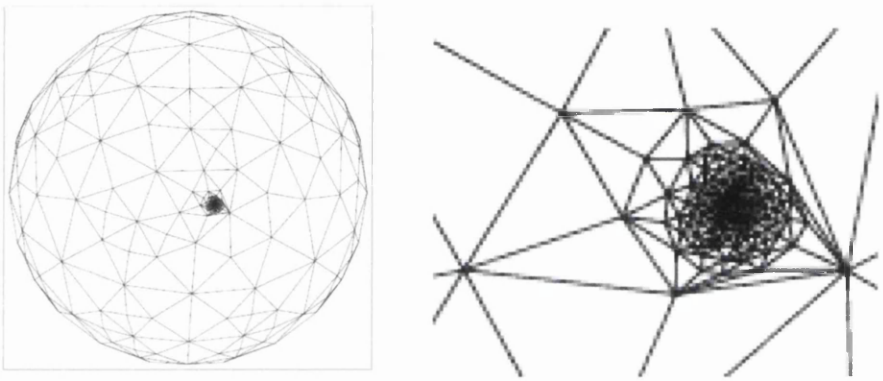


Figure 4.19 Showing sphere before gradation, note the small elements next to larger elements. A source has been used here to produce the localized elements size.

4.6.1 H-Variation

This approach is based on the H-variation measure [8] and consists of replacing a size specification by a reduced one in all directions. The reason for this is to enable the element size in the mesh to gradually change. In the isotropic case, the aim is to bound the H-variation $v(PQ)$ along an edge PQ by a given threshold value α. This will enable the user to define the threshold that they wish to work to, by setting α.

$$v(PQ) = \frac{|h(Q) - h(P)|}{\|PQ\|} \leq \alpha \qquad (4.6.1)$$

Where h(Q) and h(P) are the size specification at the points P and Q, and $\|\overrightarrow{PQ}\|$ is the length of the edge. The size specifications are then adapted to involve α (where the default is 0.35). This is done by,

$$h(P) = \min(h(P), h(Q) + \alpha\|\overrightarrow{PQ}\|),$$

$$h(Q) = \min(h(Q), h(P) + \alpha\|\overrightarrow{PQ}\|). \qquad (4.6.2)$$

The variable α has control over the outcome of the final mesh. The smaller the value of α the greater the impact the gradation procedure will have on the whole domain. If the value of α is kept larger then the effect of the gradation is kept localised. This effect can be seen in Figure 4.20, where varying values of α are shown on a surface mesh of a sphere which is locally refined using a point source.

4.7 Mesh Quality

The quality of the resulting mesh is very important for the mesh to be used in finite element calculations. As mentioned in Chapter 2, mesh smoothing and swap diagonal techniques are used to improve the quality of the mesh. In this chapter we will look at the different ways of applying these mesh improvement techniques to the mesh generation process.

- *Local Modification*

  This applies the smoothing and swap diagonal techniques locally to the mesh. The first method considered was to apply modification after every time an edge was split or collapsed (Figure 4.21). This method was extended to only apply mesh modification after a set number of split and collapse processes have taken place.

- *Global Modification*

  The mesh improvement techniques are only applied at the end of the mesh generation process (Figure 4.22), when the mesh complies to the metric.

After applying the mesh quality measures that were introduced in Chapter 3 for both the local and global measures, the results were:

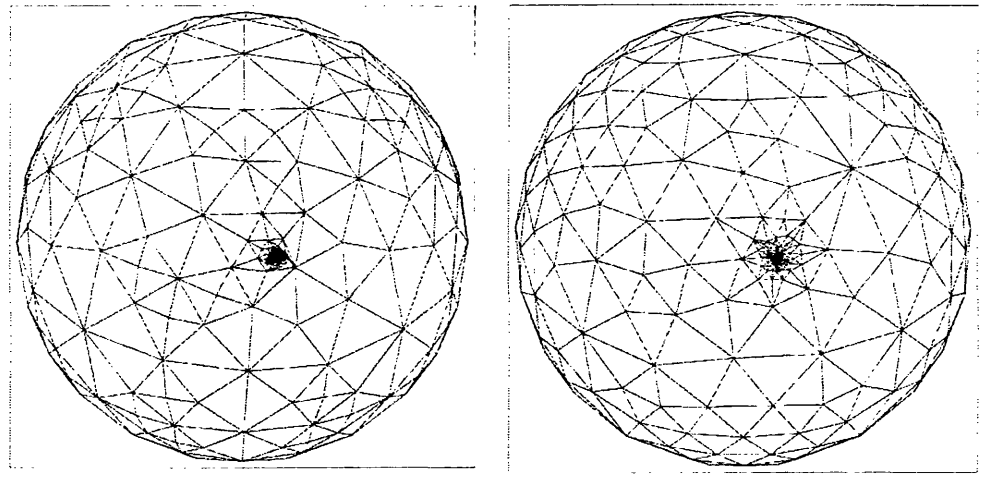
Figure	No. elements	No. points	$Qual_{avg}$	$Qual_G$	$Qual_{per}$
4.21	7241	3021	1.30	3.10	95.30
4.22	7321	3097	1.31	3.12	95.27

It can be seen that there is no real noticeable difference between the two methods. Therefore, the global modification process was chosen to improve the mesh quality. The process is applied after the mesh has been generated to comply with the metric and again after the mesh has been graded.

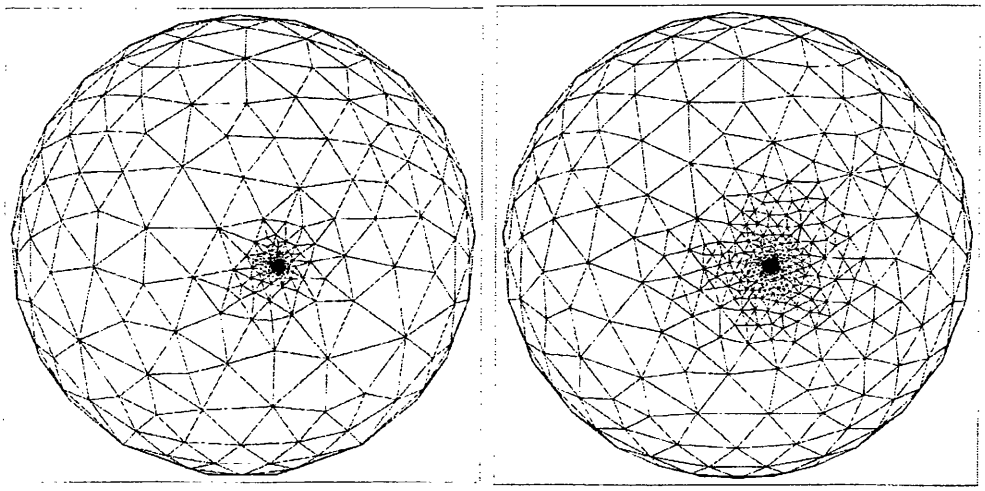
4.8 Patch Elimination

This section relates to the work in the previous chapter. In Chapter 3 mesh improvement was achieved by merging together neighbouring patches to obtain one Super Patch, where the surface definition was available. If the surface definition is not available and all that is given is the facet information, then the method introduced in this chapter can be used to obtain the same results as that shown in Chapter 3.

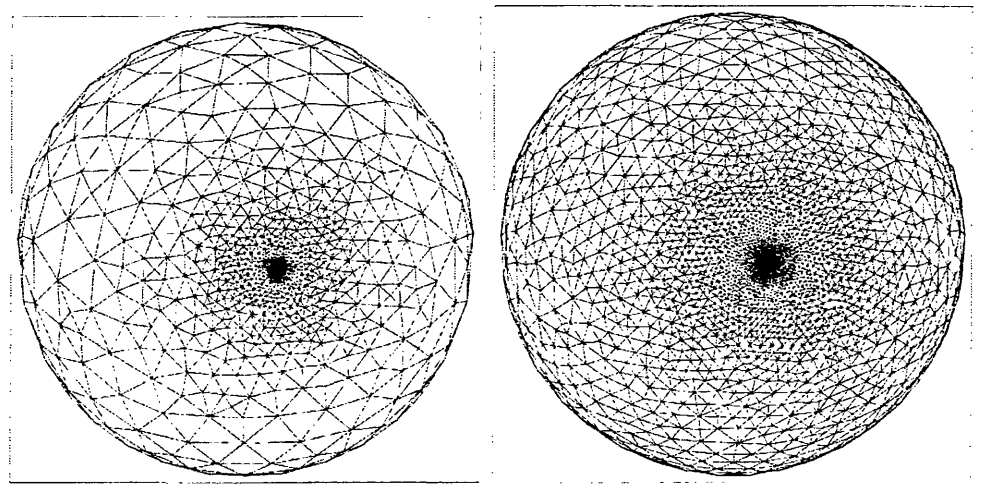
The algorithm described below will reproduce the mesh so that the small and narrow elements will be removed, by utilising techniques such as splitting, collapsing, smoothing and swapping.



(a) original mesh (b) $\alpha = 0.7$



(c) $\alpha = 0.5$ (d) $\alpha = 0.3$



(e) $\alpha = 0.2$ (f) $\alpha = 0.1$

Figure 4.20 Example of gradation.

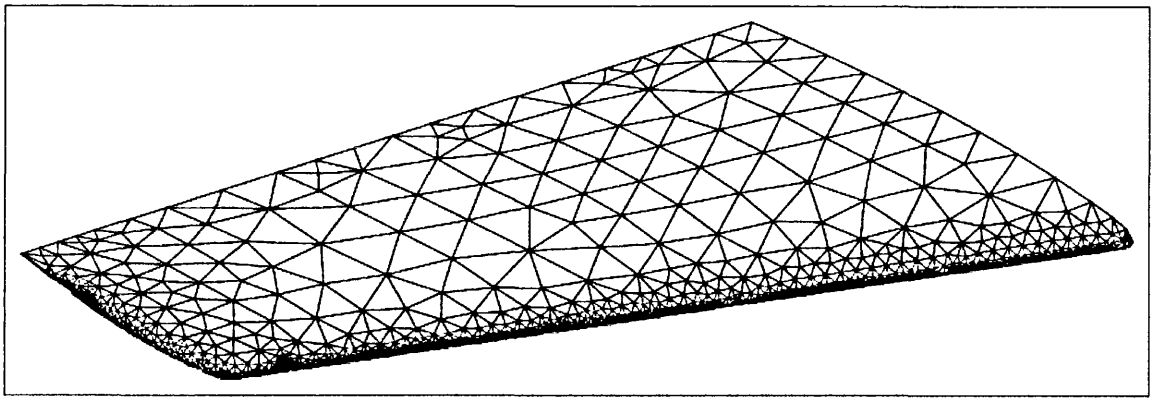


Figure 4.21 Local mesh modification application.

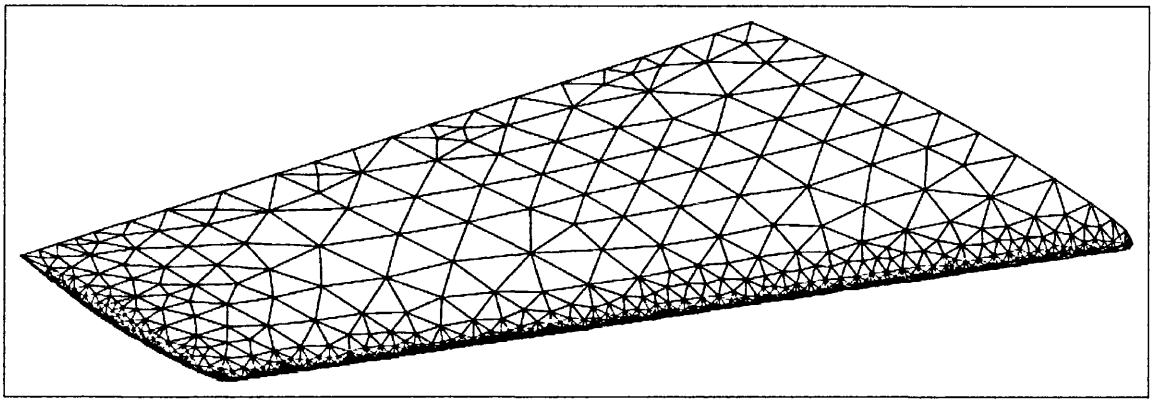


Figure 4.22 Global mesh modification application.

Figure 4.23 shows the original mesh of the gulf-stream aircraft. The original mesh shown has been generated using a set of conforming patches from a CAD system. The geometry has been generated by a designer, and one of the patches which can be seen in the original mesh is a small triangular patch which has caused small elements to be generated next to larger elements. This is the same problem as described in Chapter 3, which was overcome by the Super Patch method. The following algorithm shows how the problem can also be overcome by metric controlled meshing.

Algorithm

1. Compute l_{AB} for every edge in the mesh, where $l_{AB} = \sqrt{\overline{AB}^t G_3 \overline{AB}}$, $\overline{AB}$ is the vector of the edge being considered and G_3 is the metric as described earlier.
2. Arrange the edges in order of length.

3. Starting with the largest length first.

   If $l_{AB} \leq 1/\sqrt{2}$ then collapse edge by replacing the edge being considered by a node placed at the centre of the edge using a G_1 approximation.

   If $l_{AB} \geq \sqrt{2}$ then split the edge by splitting the two elements either side of the edge into four edges where the new point is placed using a G_1 approximation.

4. Update the edge list.

5. If there are still edges available, go to 3.

6. Swap diagonal: This technique is applied in the physical space and changes the connectivity's of nodes without changing their position. This process requires looping over all of the edges in the mesh, excluding ridge edges. Swap is to take place if the minimum angle occurring in the new configuration is larger than in the original one.

7. Smoothing: This procedure alters the position of the node without changing the topology of the mesh. The idea used is that every edge is considered as a spring of stiffness proportional to the length of the side. The node considered is moved until the spring system is in equilibrium.

8. Apply gradation on the mesh to improve mesh quality.

4.9 Examples of Patch Elimination

This section will look at a number of examples of the process outlined in this chapter using metric controlled mesh generation, gradation, mesh quality techniques and sources. Firstly, recall previous definitions. The variable ε defines the ratio in which the edge can deviate from the surface. The variable α controls the gradation of the mesh and μ is the angle used to specify a ridge edge.

Table 4.1 shows the quality measures as outlined in Chapter 3 where $qual_{avg}$ is the average value, $qual_G$ is the maximum value and $qual_{per}$ is the percentage of elements that are of quality better than 2. The quality value will

range from 1 to ∞, where the optimal value for an equilateral triangle is 1. Table 4.2 shows the quality measures related to length, area and angle, where $l_{\min}$ and $l_{\max}$ are the minimum and maximum edge lengths, l_{ratio} is the average ratio between the maximum and minimum edges that meet at a point. The $area_{ratio}$ is the average ratio of elements that share an edge throughout the mesh and finally $ang_{\min}$ and $ang_{\max}$ represent the minimum and maximum angle throughout the mesh.

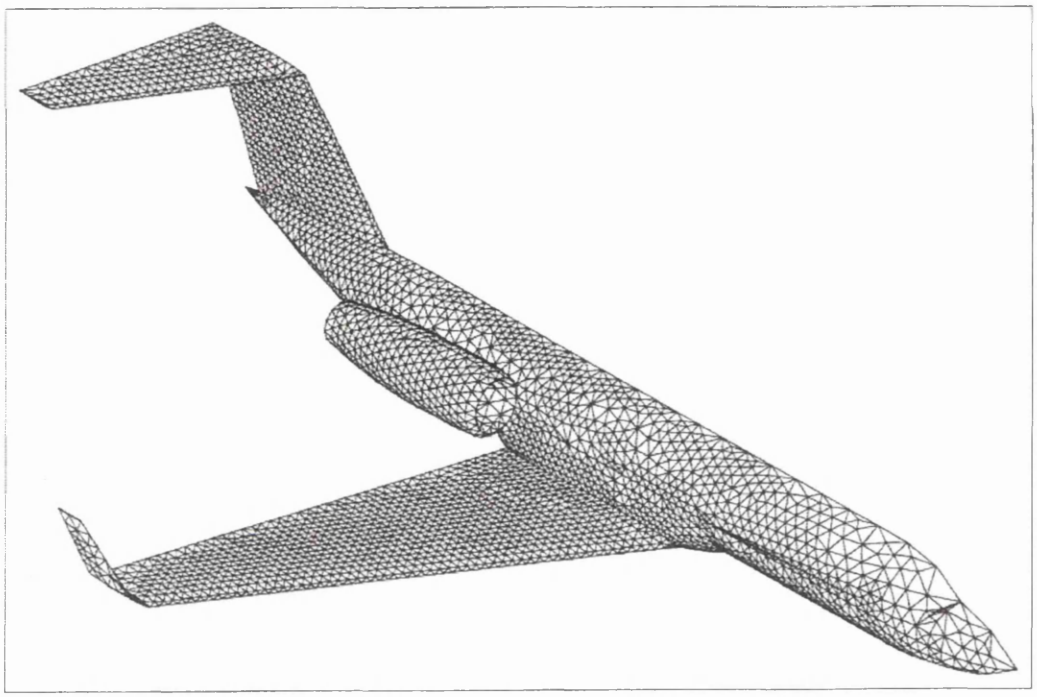


Figure 4.23 Original mesh Gulf-Stream.

The curvature based mesh generation process will overcome this problem by collapsing or splitting elements until the edge length complies to the metric. This means that the small elements shown in Figure 4.23 will be removed from the final mesh as shown in Figure 4.24, where the mesh has been produced using $\varepsilon = 0.25$, $\alpha = 0.35$ and an angle of 60 degrees to define the ridge points. Line sources have been applied to the trailing edge of the wing and tail to produce the final finite element mesh. A close up on the area of interest can be seen in Figure 4.25, where (a) is the original mesh and (b) shows the curvature based mesh.

Figure	No. elements	No. points	$Qual_{avg}$	$Qual_{G}$	$Qual_{per}$
Original gulf	12158	6081	1.49	5.86	90.3
Metric Gulf	28250	14127	1.47	5.79	92.1
F16 original	38722	19363	1.48	5.56	91.4
F16 metric	46864	23434	1.44	5.55	92.6
A3XX original	45414	22705	1.60	9.40	89.98
A3XX metric	45513	22725	1.56	6.53	91.9
Head original	1748	857	2.10	11.3	89.6
Head metric	6986	3476	1.83	8.9	90.1
Falcon original	9784	4892	1.41	3.81	95.3
Falcon metric	58582	29291	1.42	3.82	94.9
Leg original	15426	7721	1.52	3.66	93.4
Leg metric	26694	13355	1.51	3.65	92.9

Table 4.1 Statistics for examples.

Example	$l_{\min}$	$l_{\max}$	l_{ratio}	$area_{ratio}$	$ang_{\min}$	$ang_{\max}$
Original gulf	0.56	0.98	1.21	1.24	10.1	91.1
Metric Gulf	0.6	0.98	1.19	1.21	46.1	69.87
F16 Original	0.02	0.1	1.31	1.28	8.9	94.2
F16 Metric	0.01	0.1	1.29	1.27	47.3	68.65
A3XX Original	511.4	1532.1	1.28	1.35	7.6	145.6
A3XX Metric	511.4	1532.1	1.28	1.35	42.6	68.7
Head Original	6.32	121.08	9.844	5.61	6.14	130.2
Head Metric	6.32	10.47	1.54	1.41	32.9	74.2
Falcon Original	0.172	0.21	1.11	1.24	46.7	68.4
Falcon Metric	0.008	0.21	1.14	1.25	45.8	69.8
Leg Original	0.0095	0.012	1.263	1.261	52.1	66.2
Leg Metric	0.000095	0.012	1.294	1.265	50.9	69.1

Table 4.2 Quality measures, length, area and angles.

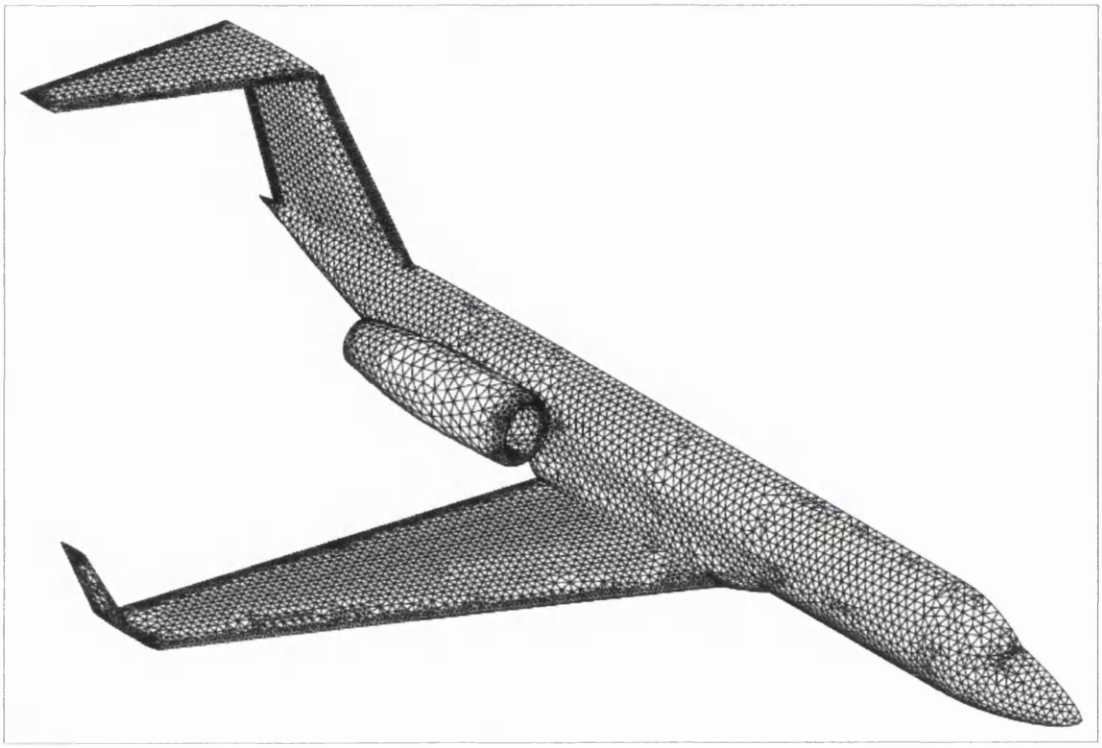
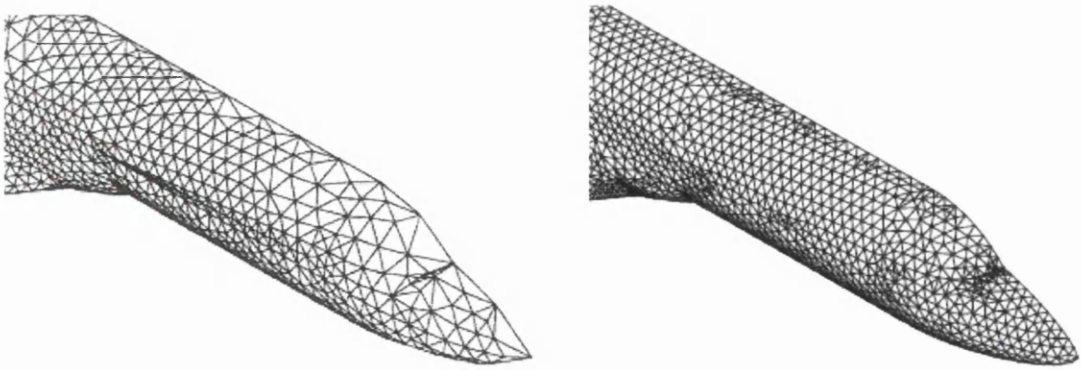


Figure 4.24 Gulf-stream showing elimination of triangular patch.



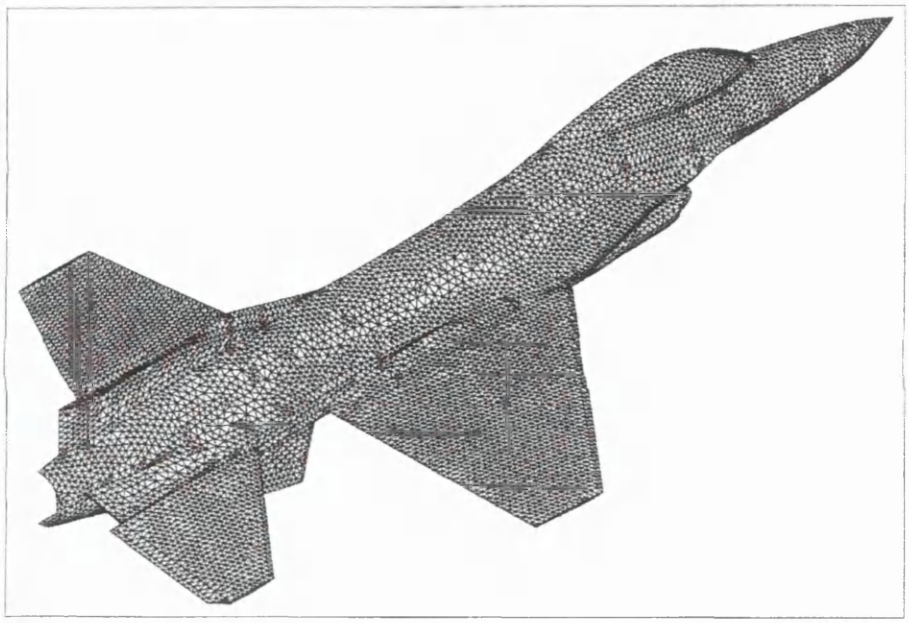
(a) Close up of original mesh

(b) Close up of curvature based mesh

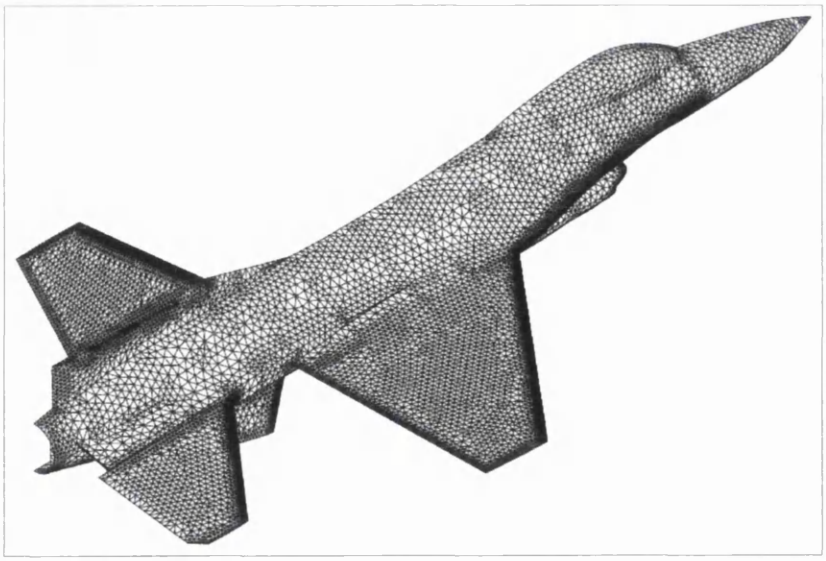
Figure 4.25 Close up of Gulf-Stream.

Another example of this method can be seen in Figure 4.26 where the F16 fighter plane has been re-meshed using $\varepsilon = 0.1$, $\alpha = 0.4$ and $\mu = 60^{\circ}$. Once again line sources have been applied to the training edge of the wing

and tail, where needed. It can be seen below (Figure 4.26) how the intersection curves from the surface patch definition have been eliminated in areas of low curvature but kept in areas of high curvature. This can clearly be seen near the tail of the plane, Figure 4.27.

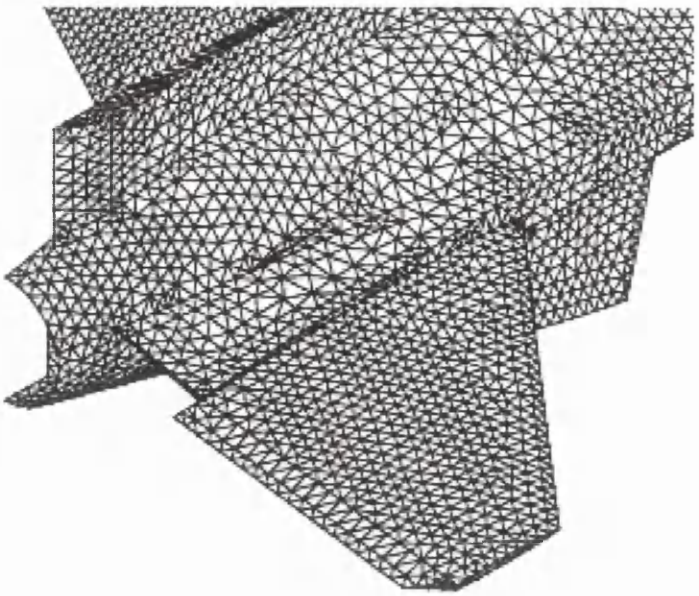


(a) Original mesh.

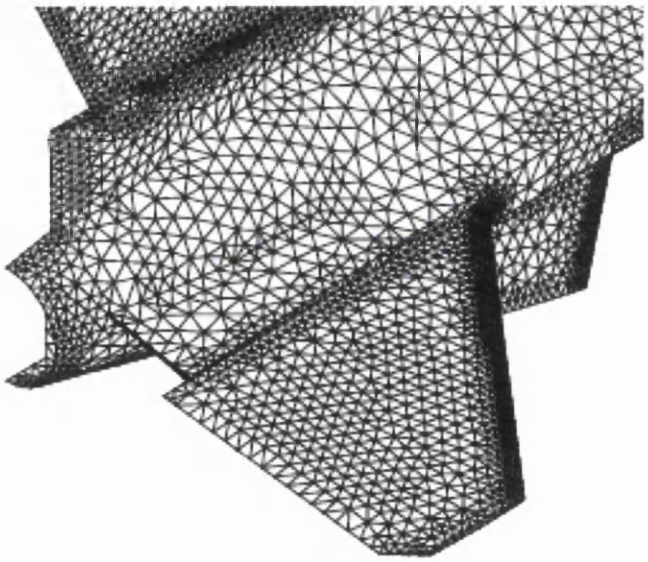


(b) Curvature based mesh.

Figure 4.26 F16 fighter plane showing original and curvature based re-meshing.



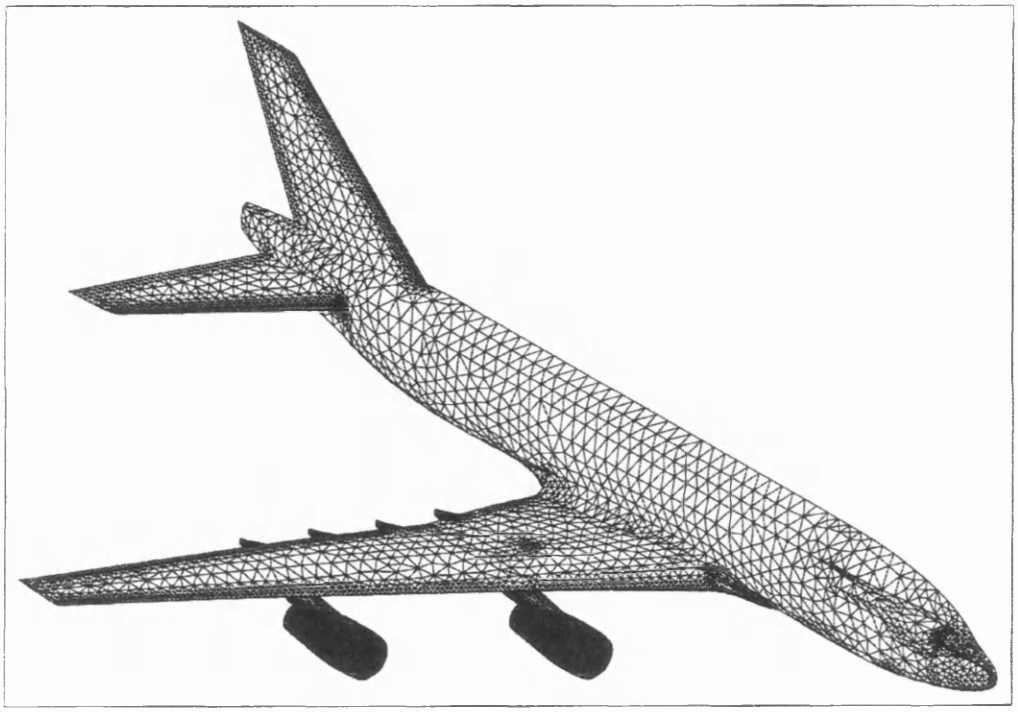
(a) Original mesh showing a close up of the trailing end of the aircraft.



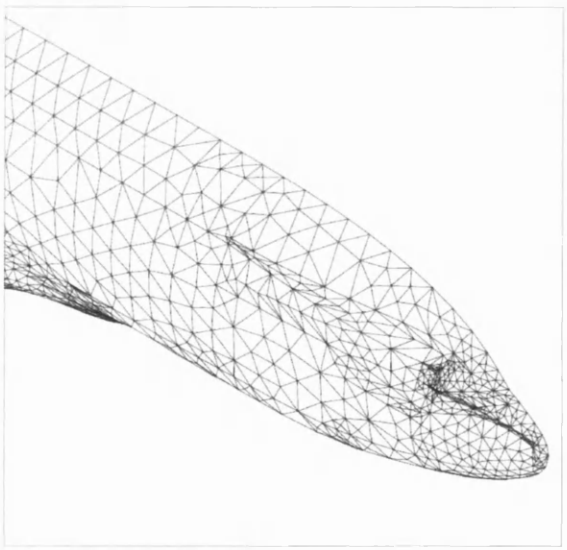
(b) Close up of the tail of the plane after curvature based re-meshing.

Figure 4.27 Close up of F16 Fighter plane.

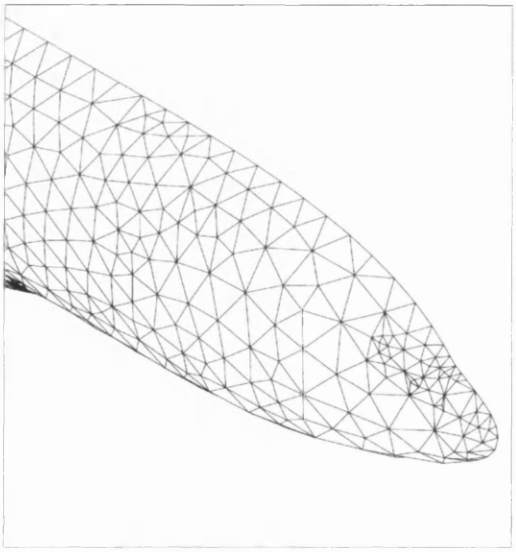
The next example, shown in Figure 4.28 is that of the A3XX aircraft where the cockpit is made up of small and narrow clusters of surface patches. The mesh was produced using $\varepsilon = 0.2$, $\alpha = 0.4$ and $\mu = 60^{\circ}$. Figure 4.28(c) shows how the badly shaped elements have been removed and the mesh has been replaced with a more constant element representation.



(a) Original mesh of the A3XX.



(b) Close up of the cockpit area on the original mesh



(c) Close up of the cockpit after of the metric re-meshing has been applied.

Figure 4.28 A3XX aircraft showing a cluster of small and narrow surface patches at the front of the aircraft.

4.10 Further Examples of Metric Controlled Meshing

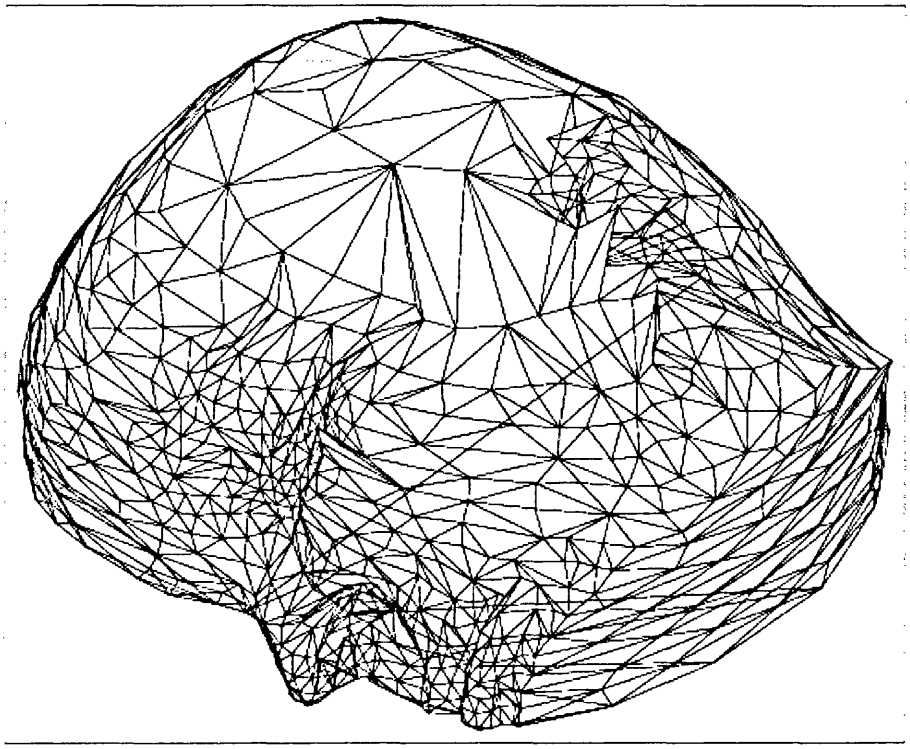
The main aim of this process is to achieve the results of the previous section. This section will look at some further applications of the method.

4.10.1 Size Specified Mesh Generation

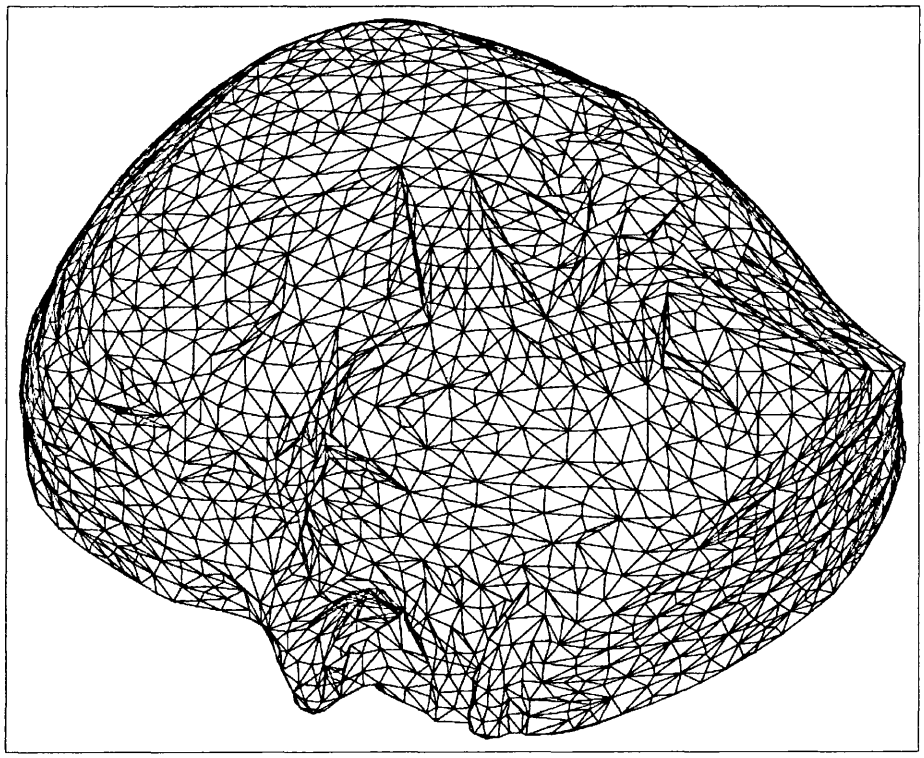
This section is related to the metric shown in equation 4.2.27, section 4.2.5., where the mesh is generated to comply with a uniform size *h*. This process is shown in Figure 4.29 where the original mesh was taken from a set of contours where the vertices are specified in each individual contour layer. The mesh has then been re-meshed to comply with the metric where the edge length has been specified. It can be seen that the mesh produced a superior mesh for finite element calculations.

4.10.2 Curvature Based Mesh Generation

This section will look at curvature based mesh generation related to equation 4.2.29 in section 4.2.5. The first example of the method is that of the Falcon aircraft as shown in Figure 4.30, where the original mesh consists of 4892 vertices and 9784 elements.



(a) Original mesh of head obtained from contours.



(b) Mesh produced from size specified metric.

Figure 4.29 Size specified metric.

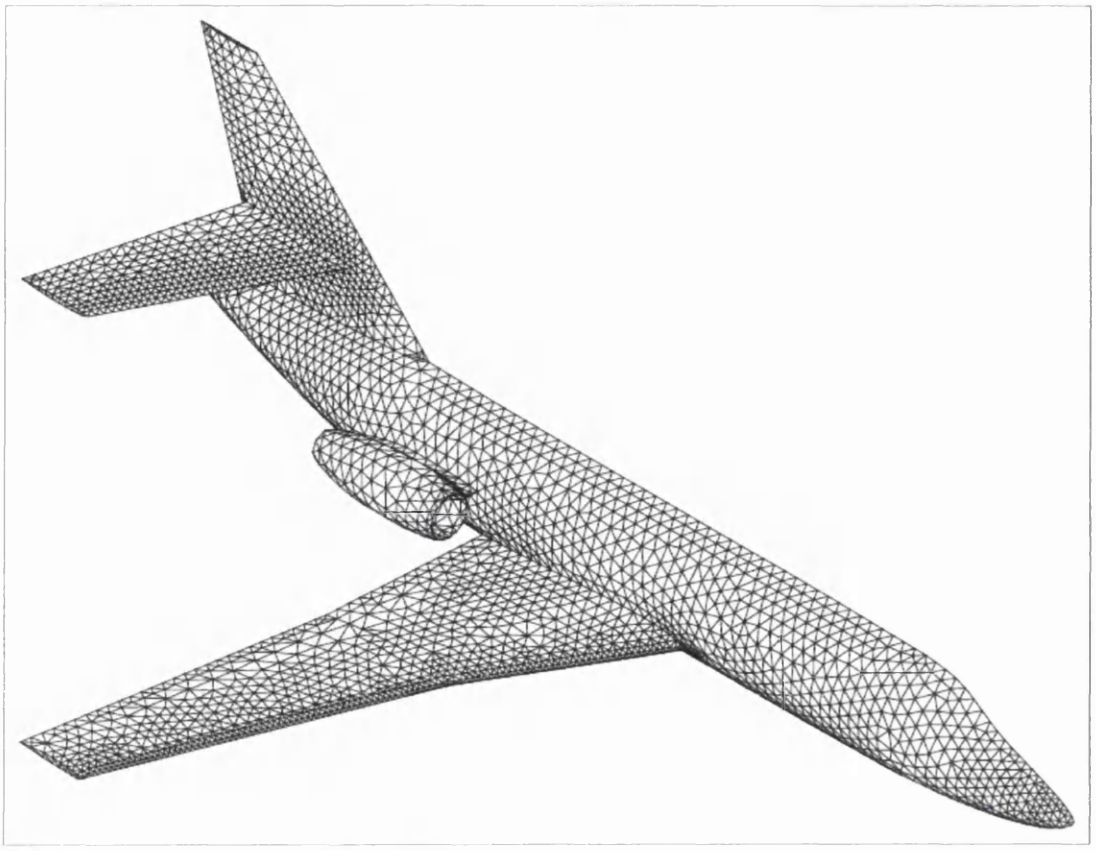


Figure 4.30 Original Falcon aircraft.

The Falcon was then re-meshed using the metric controlled mesh generation process resulting in the mesh shown in Figure 4.31. In this example $\varepsilon = 0.5$, $\alpha = 0.5$ and $\mu = 60^\circ$. It can be seen from Figure 4.31 that the original mesh has been collapsed in areas of low curvature but in areas of high curvature more elements are used to capture the curvature. This example is good for visualisation purposes where mesh quality is not important, but for finite element calculations the mesh will need to be more defined in all areas of the mesh, therefore the original mesh of the Falcon was re-meshed using $\varepsilon = 0.1$, $\alpha = 0.35$ and $\mu = 60^\circ$ (Figure 4.32). The final mesh produced 34965 vertices and 69930 elements. It can be seen that the trailing edge of the wing and tail have not been picked up as being areas of high curvature as the angle defining a ridge is set at 60 degrees. This means that all trailing edges are classed as ridges. To overcome this problem Figure 4.33 shows the Falcon aircraft after the metric controlled meshing process has been applied using sources to define the trailing edges.

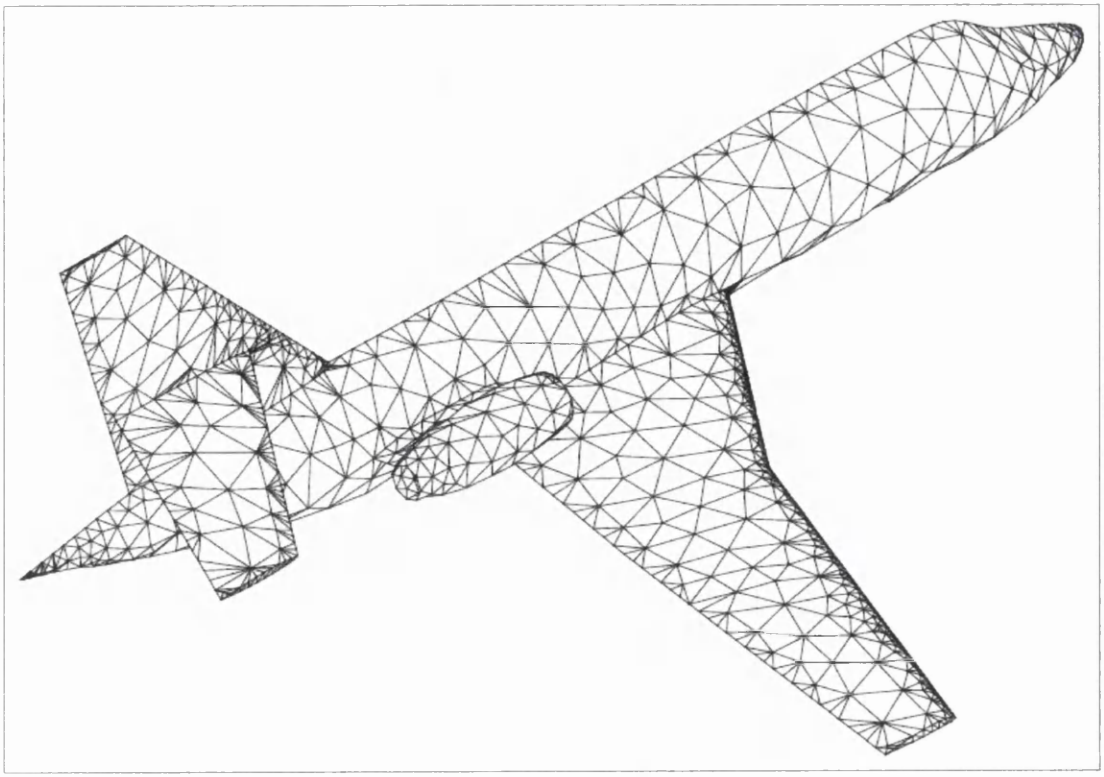


Figure 4.31 Falcon example collapse.

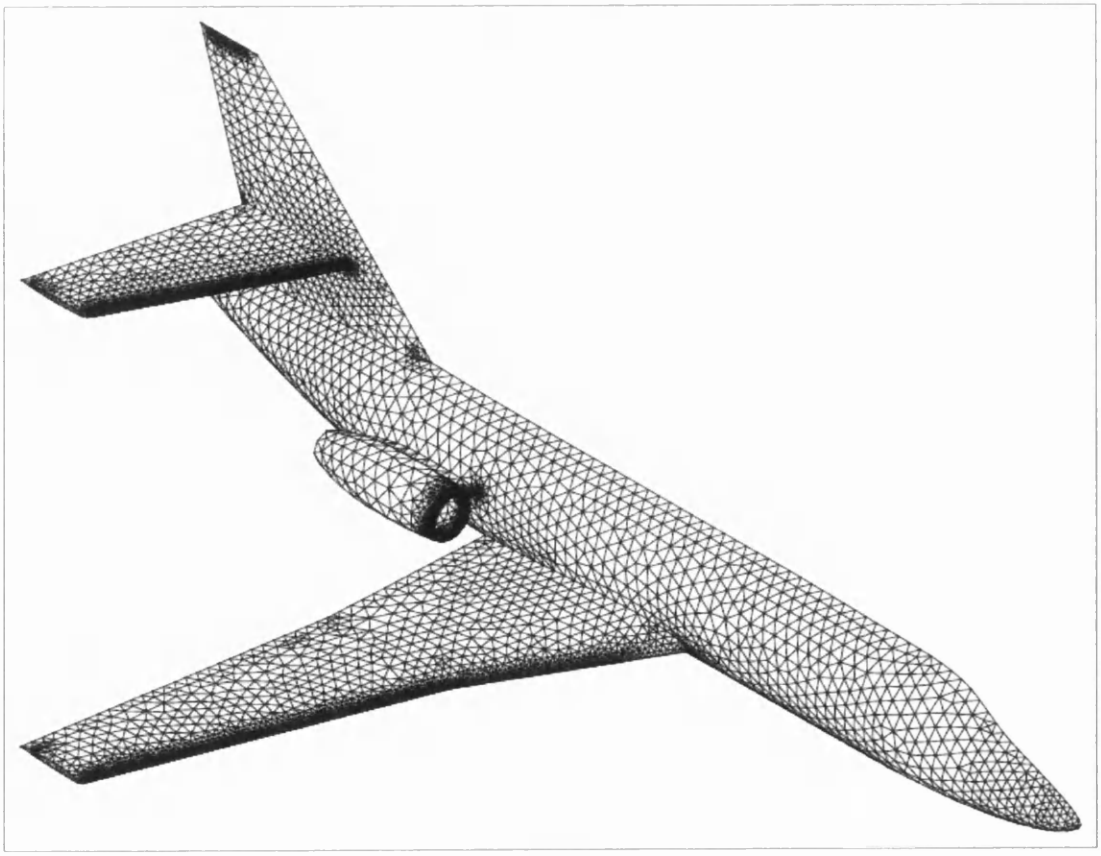


Figure 4.32 Falcon example with curvature based meshing.

For this example $\varepsilon = 0.2$, $\alpha = 0.35$, where the mesh produced 29291 vertices and 58582 elements. It can be seen from the picture that the line sources have been applied along the trailing edge of the wing, tail and tail fin to produce a mesh suitable for finite element analysis.

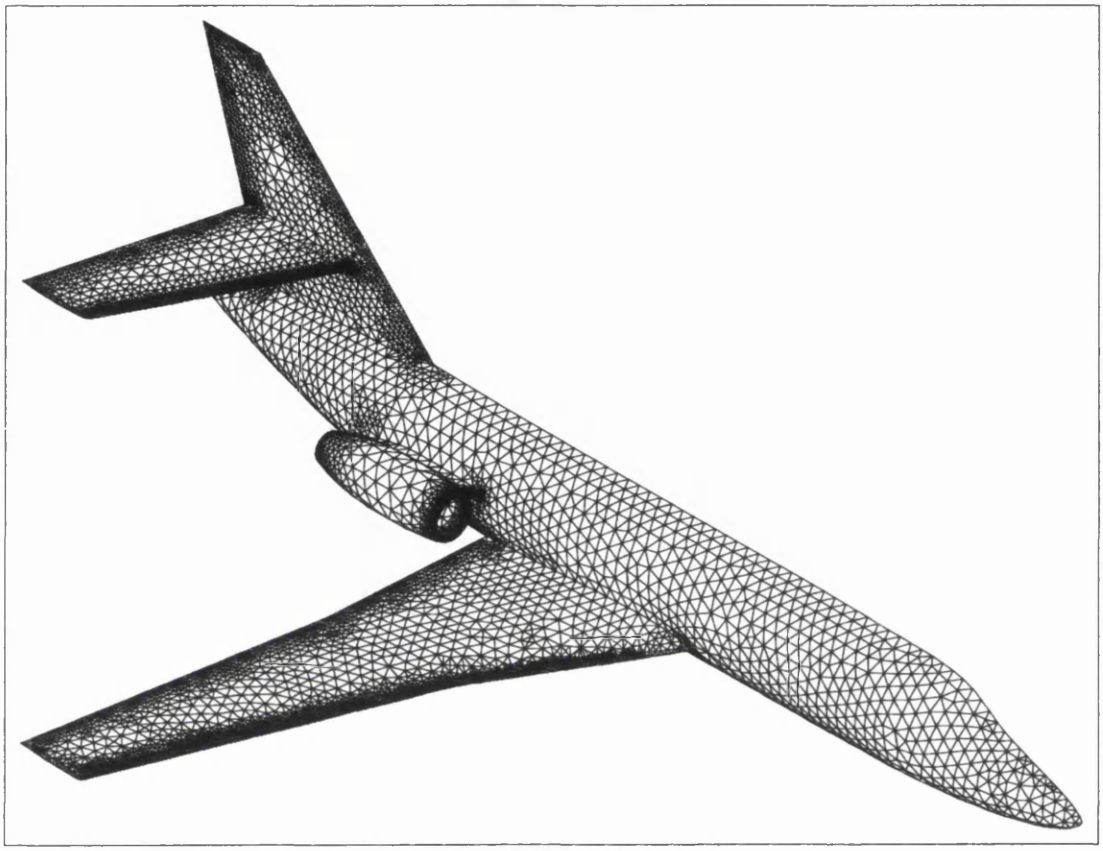
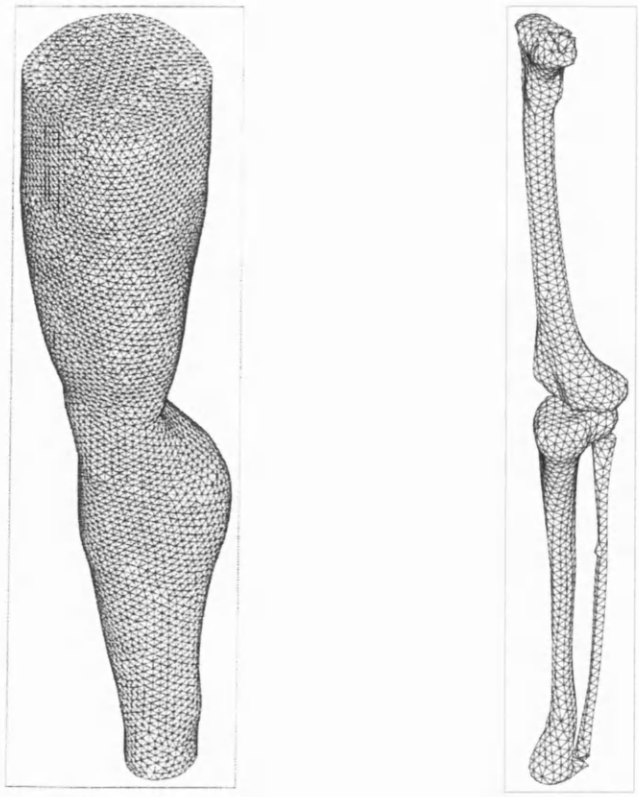
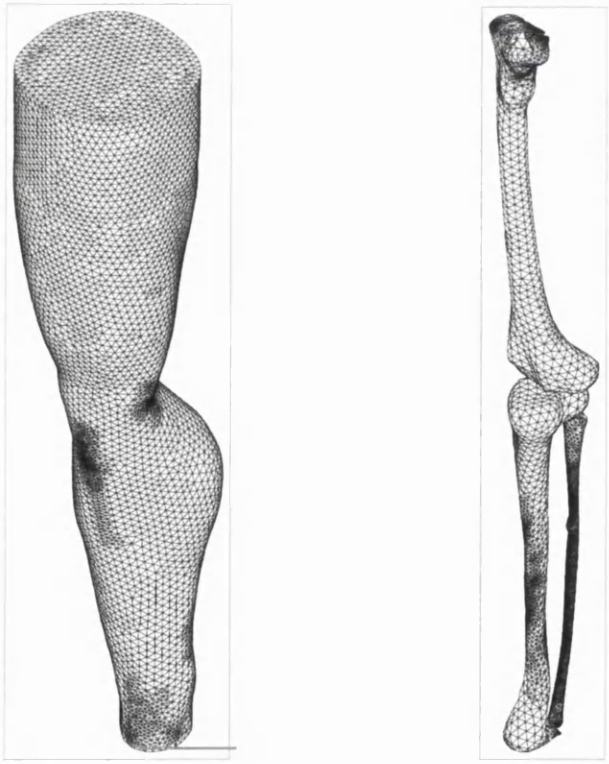


Figure 4.33 Example of falcon after metric controlled meshing with line sources.

This process can also be applied to other types of meshes such as those produced from medical imagery. In Figure 4.34 we look at a biomedical example where the original mesh has been generated using a set of contours produced from the scan of the leg - the original mesh has 7721 vertices and 15426 elements. The original mesh (Figure 4.34a) was then re-meshed producing the mesh shown in Figure 4.34b. In this example the ridges have been defined by a data file, as described in section 4.5. The mesh produced 13355 vertices and 26694 elements. For this example $\varepsilon = 0.1$ and $\alpha = 0.35$ were used.

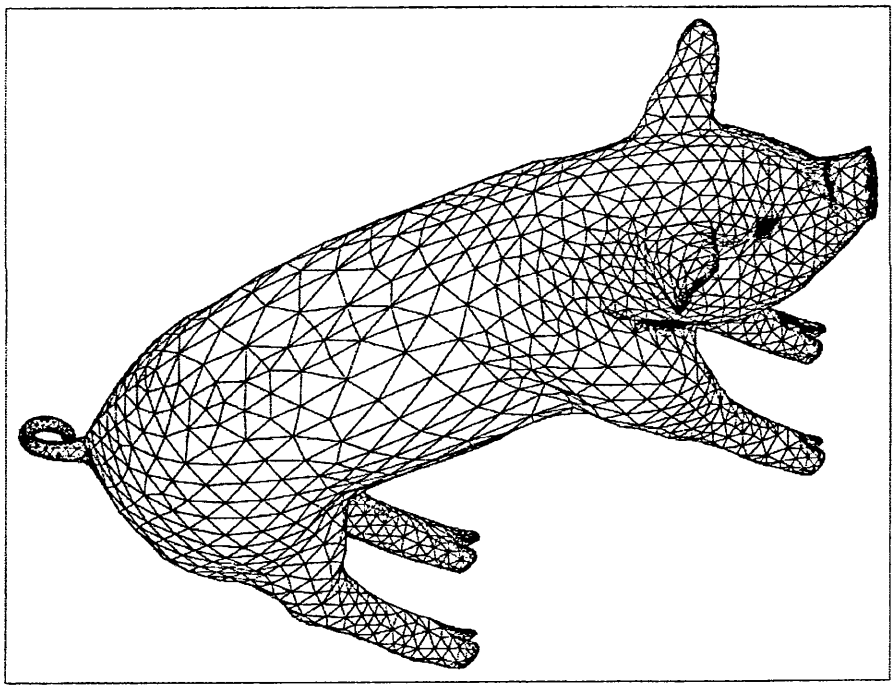


(a) Original mesh of leg.

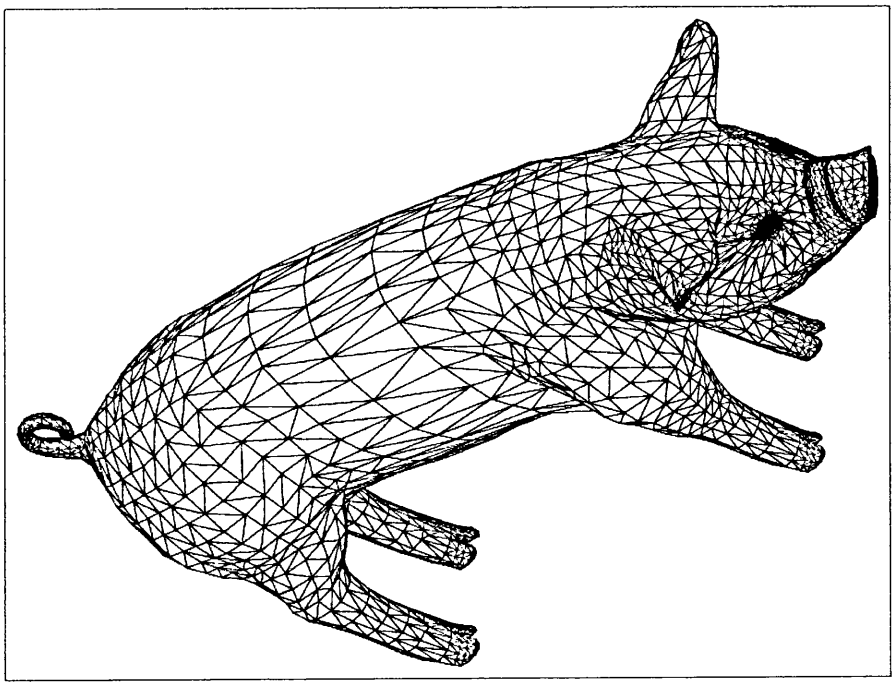


(b) Resulting mesh based on curvature.

Figure 4.34 Example of leg curvature based meshing with gradation.



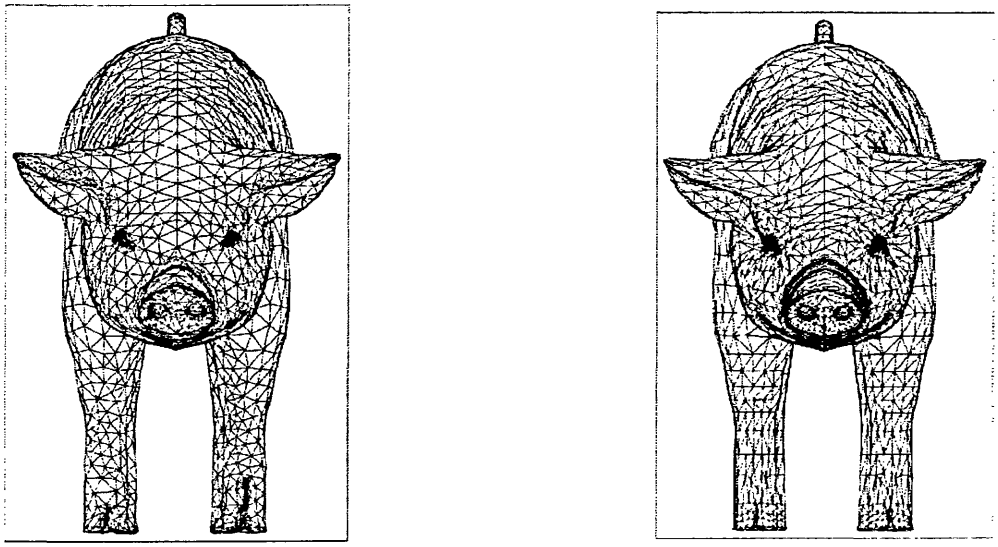
(a) Mesh of pig after re-meshing process.



(b) Original mesh of pig.

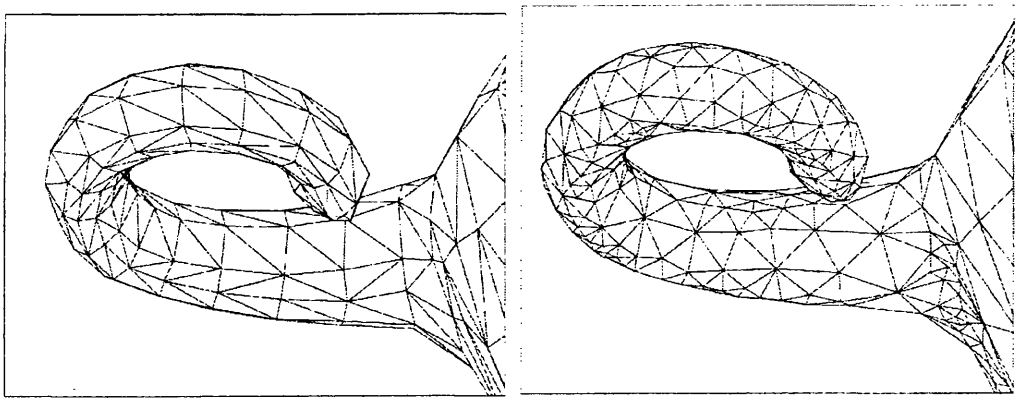
Figure 4.35 Example of a pig.

The final example is that of a pig, Figure 4.35. It can be seen how the re-meshed elements are only placed in the areas of high curvature along the snout and the tail. This can be seen in Figure 4.36 and Figure 4.37.



(a) re-meshed geometry (b) original mesh

Figure 4.36 Front elevation of the pig.



(a) original mesh (b) re-meshed geometry

Figure 4.37 Close up of the tail showing the increased elements on the tail.

5.

SURFACE MESHING FOR SCANNED IMAGES

5.1 Introduction.

Over the past twenty years there has been a growing interest in the modelling of human body parts [49-59], Bajaj looked into this area of research in 1996 [112], which has been indulged by the Visible Human Dataset [109-111]. The problem in medical imaging is in the creation of three-dimensional models from a set of tomographic cross sections. Such reconstructions of human organs are already used for;

- Surgery planning
- Prosthesis milling
- Radiation therapy planning
- Volumetric measurements

Due to the high interest in this area, there have been a great number of approaches which can be classified into two groups: surface reconstruction [49] and volume reconstruction [88]. The volume reconstruction consists of the data coming in the form of a 3D image made of voxels. The problem is then to extract and follow the faces of the voxels which are on the boundary of the objects. This is a typical situation when dealing with CT images, which has been studied by Herman [88]. In many other situations the information available for the 3D objects consists only of planar contours extracted from the cross-sectional images, a 3D structure (such as voxels) is unavailable. The problem is not a detection problem but an interpolation problem. This is where the classical solutions of Keppel [49] and Fuchs et al. [50] reduce the problem to constructing a sequence of surfaces, one between each pair of cross sections as shown in Figure 5.1. These surfaces are constructed solely from elementary triangular tiles, each defined between two consecutive points on the same contour and a single point on an adjacent contour.

The problem with this approach is that the tiling algorithms are not appropriate when there are major differences in shape or position between successive contours, also it cannot handle the case of objects with multiple contours in a single plane [49-52].

This section introduces an approach to create a 3D mesh, where the object contains multiple contours and where the number of contours varies from one cross section to the other. In order to produce a 3D mesh, 2D Delaunay mesh generation is used on each individual cross section. Unlike Fuchs et al [50] where the closest point connection is used with boundary nodes to create a surface mesh, this method looks to produce a 3D volume mesh utilising the closest point connection. The reason for this is the need to connect cross sections with multiple contours and the method of Fuchs is not capable of doing this. To aid in the computation of the 3D mesh, the idea of the medial axis is used. The reason for this is to determine an external and internal medial axis to allow an accurate connection of multiple contours. In order to produce the medial axis the Delaunay triangulation is used to produce an approximation to the medial axis. At the end of the procedure, it will be easy to produce the surface of the object by looking at the boundary of the obtained volume.

This chapter will look at utilising the ideas of Boissonnat [51-52] in order to produce a 3D surface mesh of medical imagery. From this, the mesh will be re-meshed using the approach introduced in Chapter 4.

A typical cross section for a simple geometry can be seen in Figure 5.2. This shows how the cross sections resemble slices through an object which, via the Delaunay triangulation and the closest node approach, need to be joined to produce the resulting surface geometry.

A contour line represents a cross section of constant z value and is numerically provided by the x-y coordinates of an ordered, variably spaced sequence of points on the contour line. This is not always the case, as the process could be applied to cases where the z coordinate is not constant. In this case some form of mapping has to be introduced in order to generate the mesh. Throughout this chapter cases are only considered of constant z as this is the form that medial image scans take. The total number

of contour points given depends in practice on the desired accuracy. The spacing between cross sections is assumed to be variable.

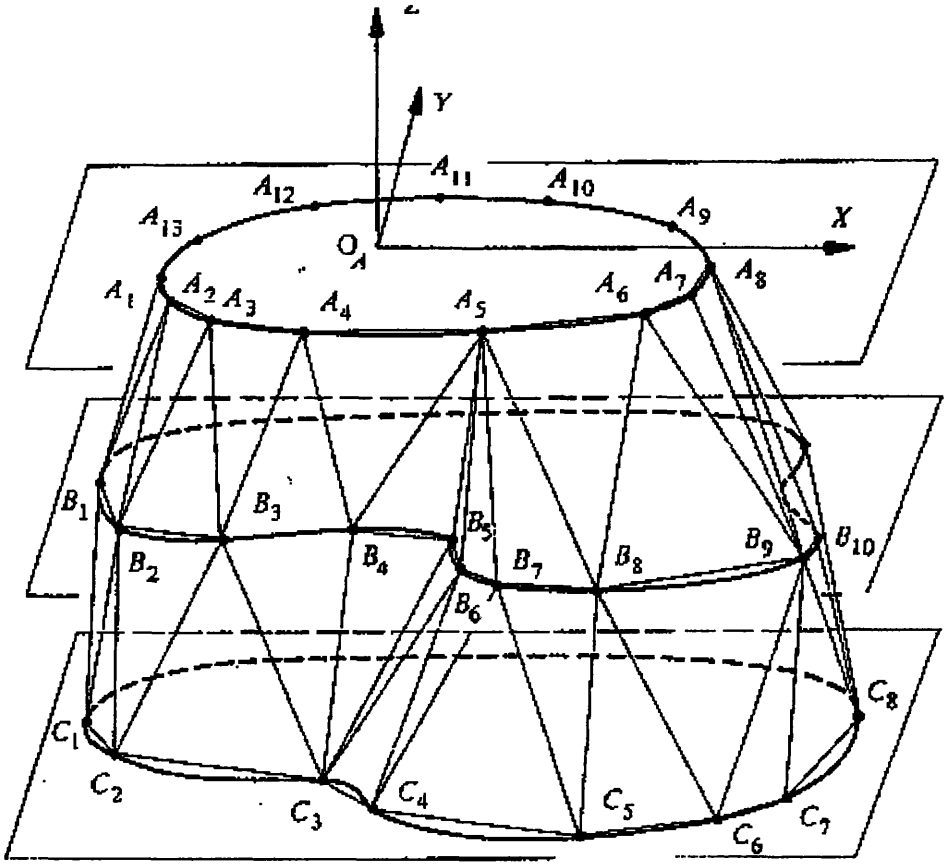


Figure 5.1 Surface reconstruction [49].

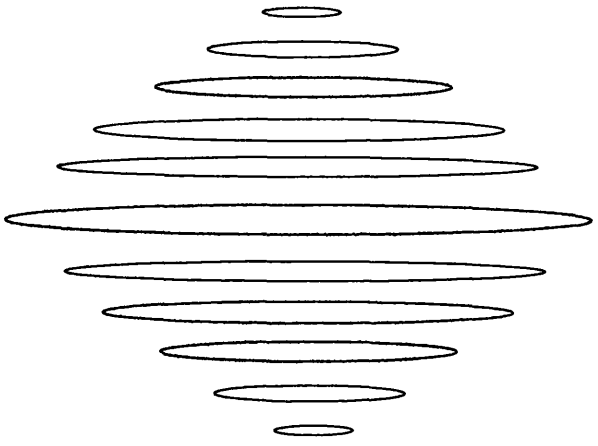


Figure 5.2 Contour layers of a simple geometry.

5.2 Algorithm

1. Read in data file, which contains the nodes on the corresponding cross sections of the geometry.
2. Generate the Delaunay triangulation on contour 1, only using the boundary points.
3. Compute the internal and external medial axis for surface 1.
4. Check if the internal medial axis crosses the boundary of the domain, if so then add more points along the boundary wherever the axis intersects the boundary.
5. Generate the Delaunay triangulation on contour 2, only using the boundary points.
6. Compute the internal and external medial axis for surface 2.
7. Check if the internal medial axis intersects the boundary of the domain, if so then add more points along the boundary wherever the axis intersects the boundary.
8. Check if the projection of the external medial axis of cross section 2 intersects the internal medial axis of cross section 1. If so then add nodes inside the domain of cross section 1 along the projection of the external axis of cross section 2.
9. Connect the two cross sections via a closest point connection.
10. Update contour list: make cross section 2 become cross section 1 and then move on to the next cross section in the layers and make this cross section equal to cross section 2. If there are no layers left then go to 12.
11. If there are layers left then go to 5.
12. Extract the surface mesh from the volume mesh that has been created.
13. Output file.

5.3 Delaunay triangulation

The construction of the contours is done by Delaunay triangulation; the reason for this is that it is closely related to the medial axis. The Delaunay triangulation is used due to all of the benefits stated in Chapter 2. The

method that is used is to subdivide the planar regions into triangles using Delaunay 2D triangulation and then extend these triangles to tetrahedra via the closest point connection, by linking them to the contours in adjacent cross sections. From this a surface mesh can be extracted which can be used for any type of computer analysis. The first task is to create a 2D Delaunay mesh on the cross sections as shown in Figure 5.3. When generating the Delaunay triangulation for the contours the swap diagonal method is not applied to the mesh so as to retain the boundary of the domain.

5.3.1 Contour containment

The problem of connecting contours means that contour lines have to be approximated by simple closed polygons. Triangulating the polygons vertices may result in a triangulation where contour segments are not guaranteed to be edges of the triangulation. In this example the triangulation has to satisfy the following requirement:

All contour segments have to appear as Delaunay edges in the Delaunay triangulation (contour containment condition).

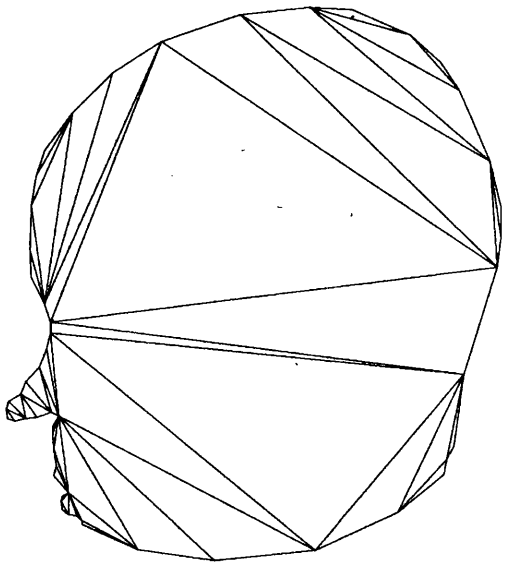


Figure 5.3 Delaunay triangulation of the cross section of the head, using only boundary nodes.

An easy solution is to insert new points on the contour edges, so that the contour shape is not changed. Figure 5.4 shows how a mesh has had a

point inserted so that the contour segments comply with the Delaunay triangulation, whilst still containing contour segments as edges of the triangulation. Figure 5.4a shows a contour where It is evident that triangle (a, b, c) cannot be part of a Delaunay triangulation since its circumcircle would contain another node. Figure 5.4b shows the Delaunay triangulation of the contour points, where the contour segments which are incident with Delaunay edges are drawn bold. The contour edge (a, b) is contained after insertion of a node on a, b, where the node is placed along the intersection of the edge (a, b and c, d,) as shown in Figure 5.4c. This is not a fool proof method but for all the cases that have been investigated, no problems have occurred.

5.3.2 Internal and External Voronoi skeleton

Once the containment condition is satisfied, the Delaunay triangles can be divided into two groups: internal triangles lying inside the contours and external triangles lying outside the contours. The two groups are identified by considering the orientation of the triangles and assuming the boundary is complete.

An Internal Voronoi Skeleton (IVS) is the edges of the Voronoi skeleton that are dual to an internal Delaunay edge (that is an edge shared by two adjacent internal Delaunay triangles). Similarly the External Voronoi Skeleton (EVS) is when the edges of the Voronoi skeleton are dual to an external Delaunay edge, see Figure 5.5, where the picture is made up of three contours each with their own internal skeleton and the corresponding external skeleton.

The Internal Voronoi Skeleton is not guaranteed to lie inside the polygon boundaries. Even worse, polygons with the same shape but different distribution of vertices may have totally different Voronoi skeletons, as shown in Figure 5.6, where the same geometry has been generated with two different distributions of vertices. The first has 8 vertices, whereas the second has 10 vertices. It can be seen how this different distribution of vertices has affected the Internal Voronoi Skeleton. Thus the quality of the shape representation is not very high at this stage

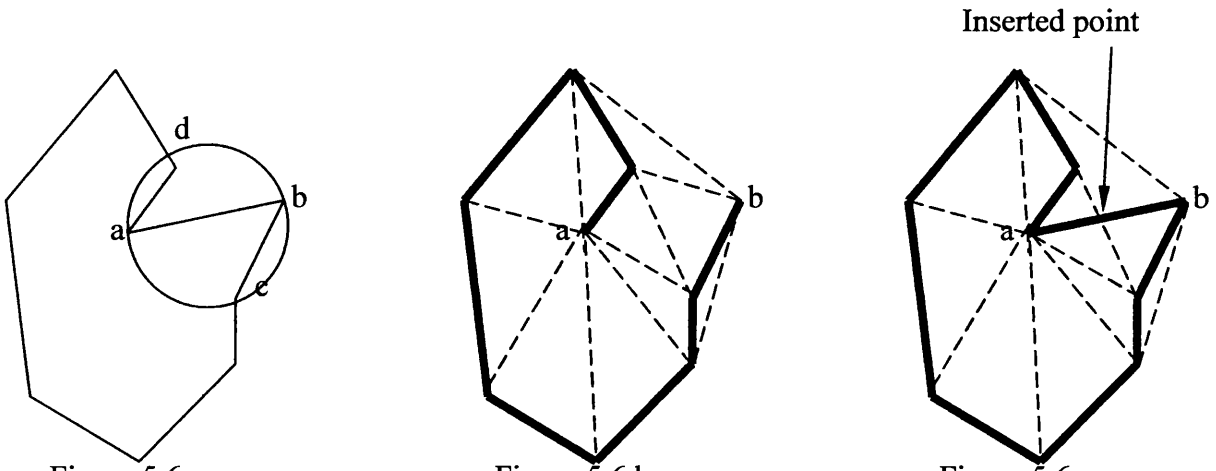


Figure 5.4: Contour containment.

5.3.3 Increasing the quality of the shape representation

To guarantee that the Internal Voronoi Skeleton lies inside its contour polygon, new vertices are added on contour segments that include obtuse Delaunay triangles, see Figure 5.6:

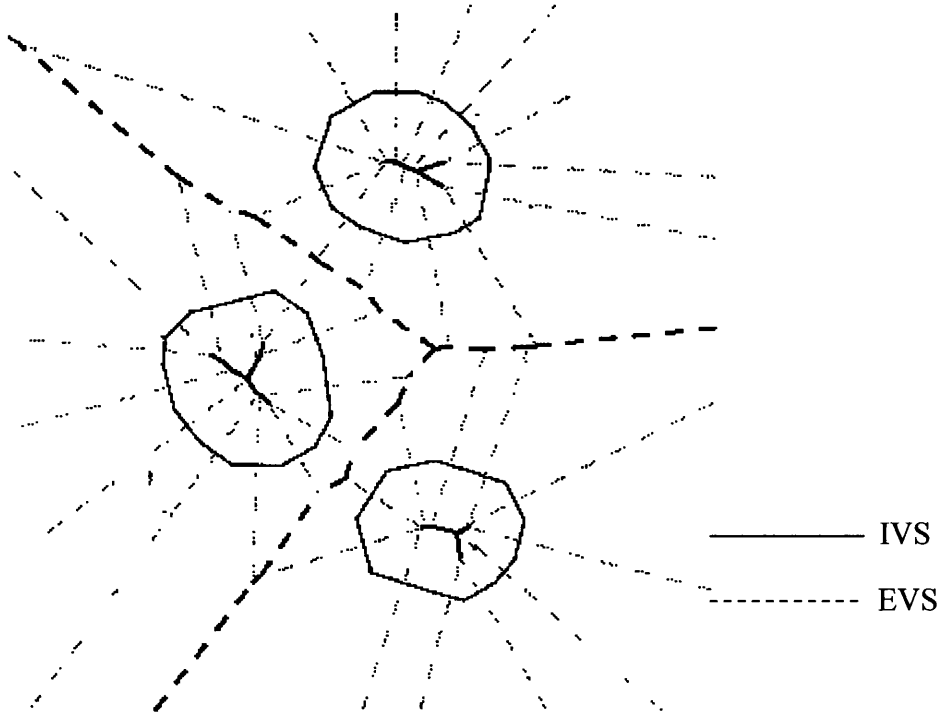


Figure 5.5 Internal (IVS) and External (EVS) Voronoi Skeleton of three contours.

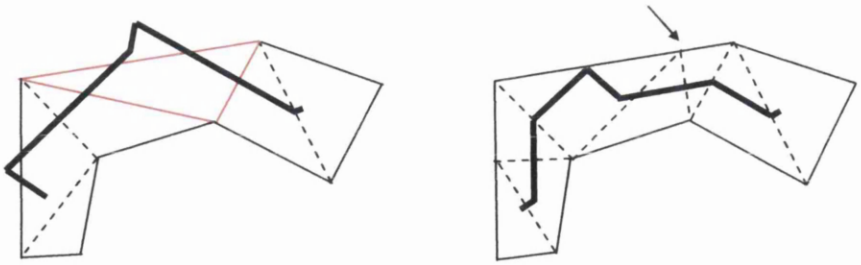


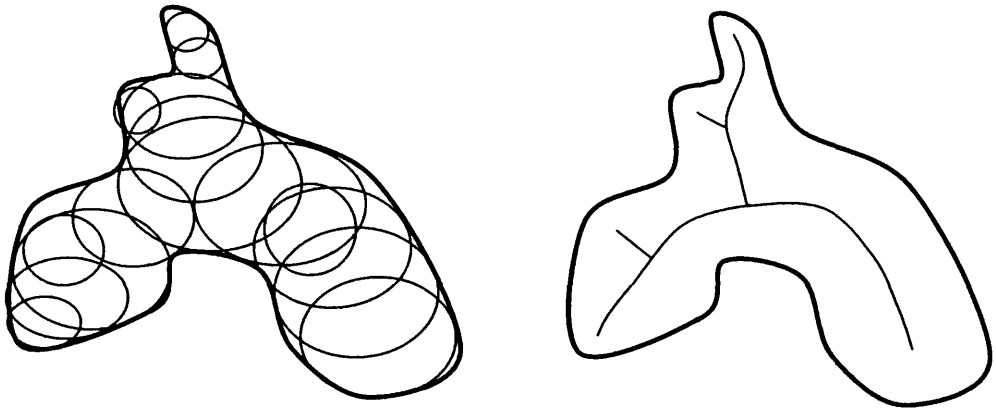
Figure 5.6 Vertices added to keep Internal Voronoi Skeleton within the domain.

For each contour segment, the opposite angle of the triangle containing the segment is considered. If it is greater than 90 degrees, its corresponding Voronoi vertex lies outside the triangle. In this case a new point is added at the normal projection of the opposite vertex onto the contour segment. This will divide the obtuse triangles in two right-angled triangles, as shown in Figure 5.6, where the original triangulation was causing the internal skeleton to lie outside the domain. This is due to the element highlighted having an angle greater than 90 degrees. In order to contain the internal skeleton within the domain a point was added on the boundary along the normal projection of the point with the angle greater than 90 degrees. The resulting internal skeleton can be seen in Figure 5.6. It is important to point out that this procedure doesn't eliminate all obtuse angles in the Delaunay triangulation, only obtuse angles opposed to contour segments are detected. But it is sufficient to guarantee that the Internal Voronoi Skeleton stays inside its contour.

5.4 Medial axis

The medial axis of a 2D region is described as the locus of the centre of an inscribed disc of maximal diameter [84]. The disc changes size to touch the boundary wherever it is within the domain (see Figure 5.7). The medial axis and the radius of the inscribed disc is known as the medial axis transform (MAT) [84-87]. Computing the medial axis is a problem, therefore, the Delaunay triangulation will be used to approximate the medial axis. The medial axis is only an approximation as the inscribed disc no longer touches

the boundary as shown in Figure 5.7, but now only intersects the boundary, through the nature of the Delaunay triangulation.



(a) Inscribed disc, (b) Medial axis.

Figure 5.7 Medial axis.

Applications for the MAT have been proposed in a wide variety of contexts [84-87]. One problem is that the MAT is hard to compute exactly. Attalli and Montanvert [83] and Sheehy, Armstrong and Robinson [85] have proposed approximating the medial axis using the Voronoi diagram.

In general, by increasing the number of vertices on the contours means that the Voronoi skeleton tends towards the medial axis as shown in Figure 5.8.

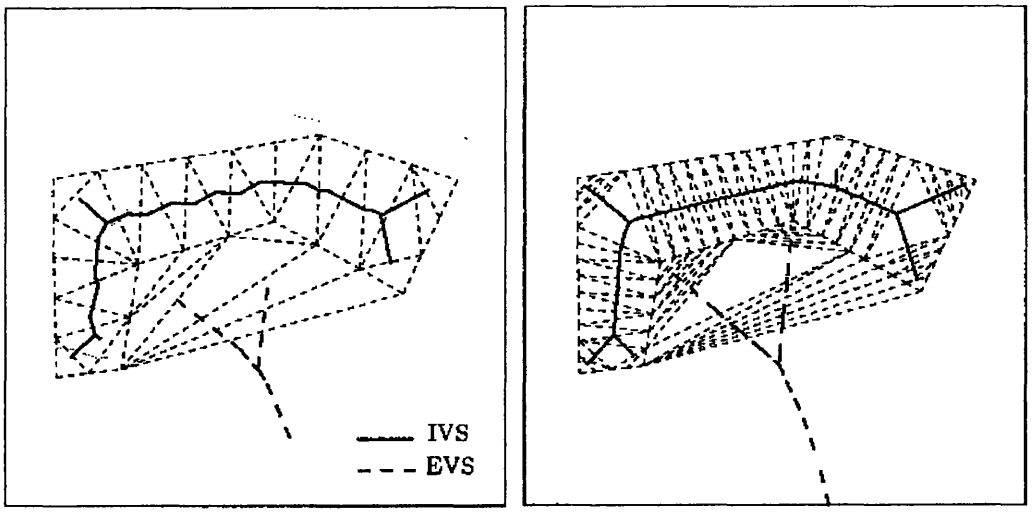
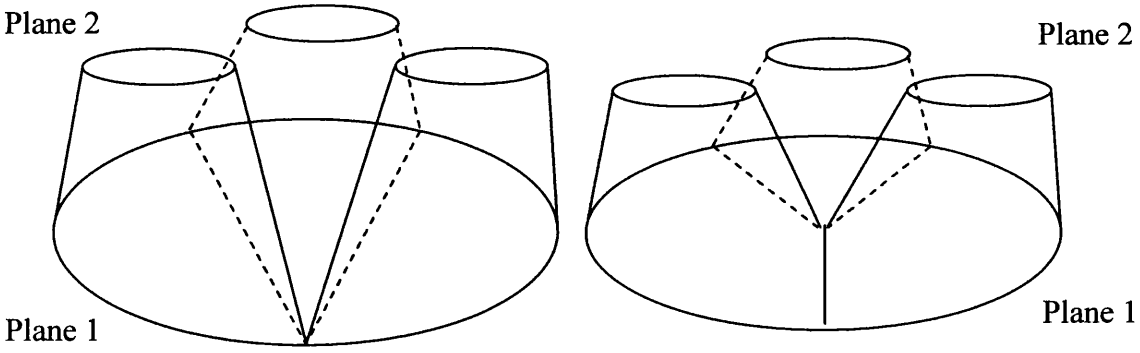


Figure 5.8 Approximation of the medial axis.

5.4.1 Introducing internal vertices

So far the method has only considered cases where each cross section only contains one contour. But, for multiply connected cross sections, where the cross section may contain a number of different contours, the representation of the geometry may be lost. In order to give a more accurate representation of the geometry, the improvement of the mesh has to be considered. The way in which this is approached is to place points within the domain. This is achieved by projecting the external skeleton in plane two onto plane one and if the External Voronoi Skeleton in plane 2 crosses the Internal Voronoi Skeleton of plane 1 then vertices are added along the projection of the external skeleton onto plane 1 within the domain. This way the final mesh that is obtained will be a better representation of the geometry. The reason for this can be seen in Figure 5.9, where plane one consists of one contour and plane two consists of three contours. Figure 5.9a shows the geometry with a solution without internal nodes and then again with internal vertices, Figure 5.11b. It can be seen how Figure 5.11b is a better representation of the geometry. The spacing of the nodes inside the domain is computed as the smallest distance between the nodes along the boundary of the domain. Then cross section one is re-meshed using the Delaunay triangulation for 2D utilising the new internal nodes.



(a) Without internal vertices. (b) With internal vertices.

Figure 5.9 Addition of internal vertices.

5.5 Volume reconstruction

The problem has been reduced from finding an overall reconstruction to computing a 3D solid between each pair of adjacent cross sections. In each cross section, the input consists of one or several closed simple polygons, which possibly may lie one inside another. The contours are oriented in such a way that the inside of the object they are describing is on its right side and the outside on its left. Let P_1 and P_2 be two adjacent cross sections, where P_1 represents plane one and P_2 represents plane two.

Algorithm

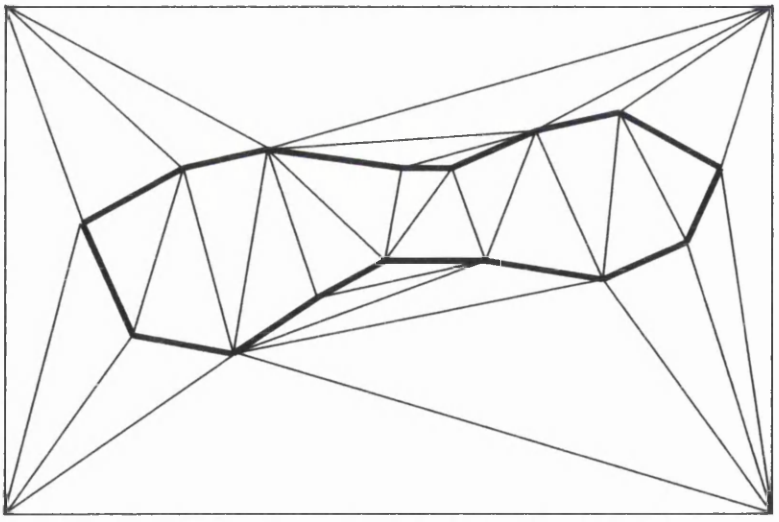
This algorithm shows how to construct a 3D mesh from a set of planes.

1. Remove triangles connected to the convex hull.
2. Eliminate the nodes of the convex hull.
3. Scale all points between 0 and 1, for ease in storing of data in the ADT.
4. Create ADT tree for existing 2D triangles.
5. For every triangle on the bottom contour, using its circumcentre find the closest point on the top layer, to form an element.
   a. Check that the edges of the element do not intersect any other element. This is where the tree is used to search within a given box to find all elements within the box.
   b. If no then go to 6, else go to 5 and find the next closest point.
6. Update the tree, add the new faces of the tetrahedra to the ADT and delete existing faces and add edges along with their corresponding nodes and angle to the link list.
7. For every element on the top contour using its circum-centre find the closest point on the bottom contour.
   a. Check that the element does not intersect any others.
   b. If not then go to 8, else go to 7 and find the next closest point.
8. Update tree and add edge and corresponding information to the link list.

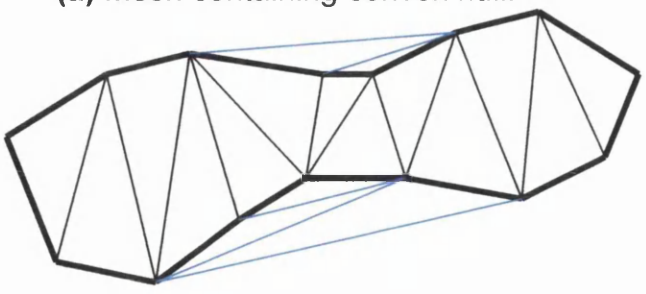
9. Look to close the domain. Since the elements so far have been created containing either a face in plane one or a face in plane two, which will leave the elements that have two nodes in plane 1 and two nodes in plane 2, to be generated. This will ensure that the domain in closed.
10. Remove elements which lie on the outside of the domain. This will mean that the elements will lie on the external medial axis.
11. Remove non-solid connections. Even after removing the elements outside of the domain there may still remain non-solid connections, which need to be removed.

5.5.1 The subdivision of the plane

As mentioned in Chapter 2, the Delaunay triangulation is contained within a convex hull. The first task in the construction of the mesh is to remove the elements connected to the convex hull, and also to remove the four nodes of the convex hull.



(a) Mesh containing convex hull.



(b) Mesh after removal of convex hull (blue=external elements).

Figure 5.10, Removal of convex hull

This will leave the remaining mesh containing internal and external elements, where all external element nodes are connected to the boundary of the domain, see Figure 5.10. The next process is to subdivide the planes P_1 and P_2, which is done by using the Delaunay triangulation on the contour vertices of each plane, see Figure 5.11. The Internal Voronoi Skeleton is checked to see if it crosses the domain. If so then more vertices are added onto the domain to retain the shape representation as described in the previous sections.

5.5.2 Construct a 3D solid

This section aims to produce a 3D solid using 2 consecutive cross sections via a closest point connection. A solid is defined as any limited portion of space bounded by surfaces. Among the simplest solids are the sphere, cube, cone and cylinder [101].

Before this we have to scale all points on both planes between 0 and 1. The reason for this is for ease in the storing of data in the ADT as mentioned in Chapter 2. The ADT is an efficient method of storing and searching for information. This method utilises the tree to store faces.

In order to achieve the 3D connection the closest point connection is utilised, where for each triangle in plane 1 the vertices in plane 2 are searched to find which one lies closest to the circumcircle of the triangle being considered to form an element. The same is done with plane 1 and plane 2 reversed, see Figure 5.12. Once a closest point has been chosen the element needs to be checked so that it doesn't intersect with any other elements within the domain. The check is made by obtaining a box that surrounds the proposed element and the tree is searched to form a list of faces that lie within the box.

Once the list of faces has been compiled, the edges of the faces within the box can be checked against the proposed element and any intersection can be flagged. If there is none then the element is created and the tree is updated. If there is an intersection, then go to the next closest point and check again, until an element can be created.

Up until now the triangulation consists of tetrahedra, which are called type T_1 or T_2, depending on whether they have a facet in P_1 and a vertex in P_2, or a facet in P_2 and a vertex in P_1. The third type of tetrahedra, called T_{12}, has one edge in P_1 and one edge in P_2, these are created by filling the remaining mesh, whilst checking for a solid geometry. The result is a filled domain of three types of tetrahedra.

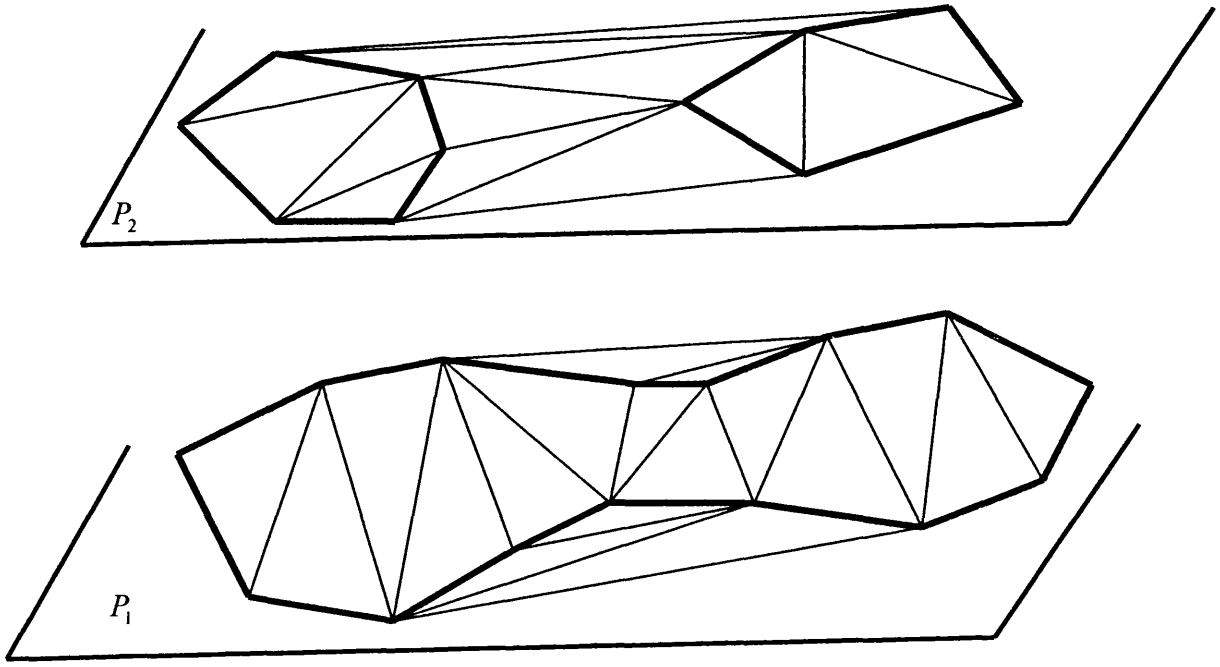


Figure 5.11 Delaunay triangulation of contours in two adjacent planes.

The Figure 5.13 shows two plane squares containing 2 triangles which are being considered to be joined to create a cube. The first process is to create the T_1 and T_2 elements as shown in Figure 5.13b and Figure 5.13c respectively. Then the domain needs to be closed with the remaining element type T_{12}.

The way in which this is done is that throughout the generation process whenever an element has been created a linked list is formed which will contain an up-to-date list of the edges that are available, containing the two nodes either side of the edge and its corresponding external angle as shown in Figure 5.13d. Every time a new element is created the list is updated and the new angle computed.

When closing the domain the angle is examined and if it is less than 180 degrees then the element is closed. A check for intersections is conducted and if none are found then the element is closed and the linked list and ADT updated.

An example is shown (Figure 5.13e) where the edge 3-5 has nodes 4 and 8 either side of it and the external angle is less than 180 degrees, with the aim to close this element. Once the element is closed the linked list and ADT are updated. The edge also has another two nodes (2 and 6) on either side, so once again a check for closure needs to be performed.

An edge can be duplicated if the nodes of the two adjacent faces are different to the existing configuration in the list.

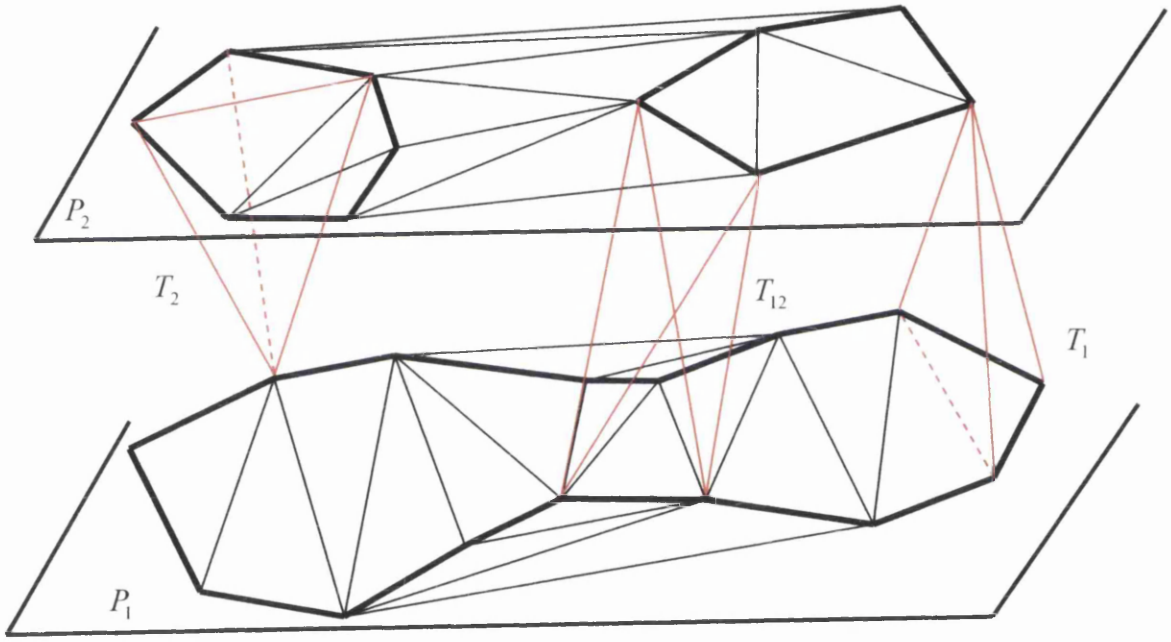
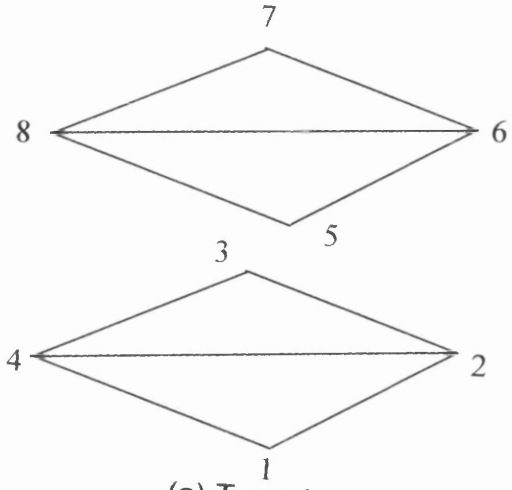
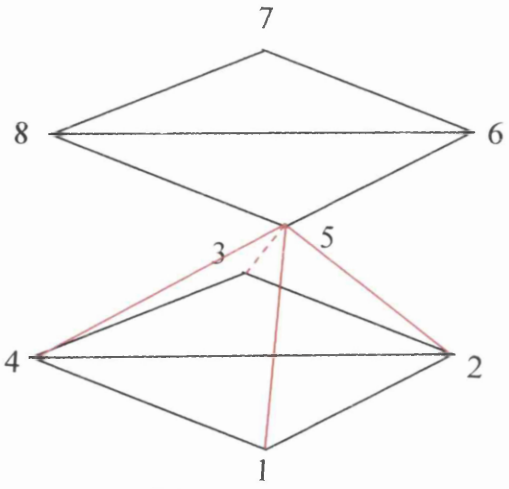


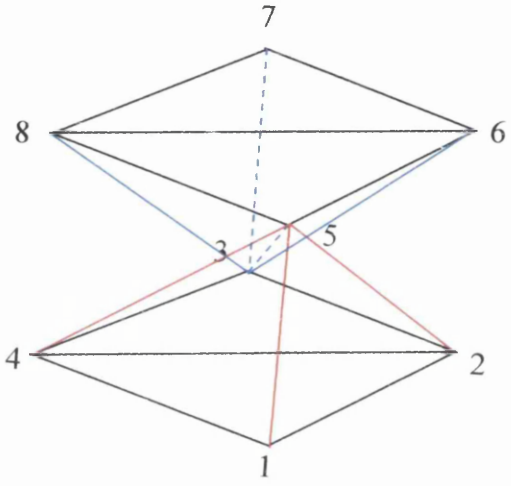
Figure 5.12 Creating tetrahedra.



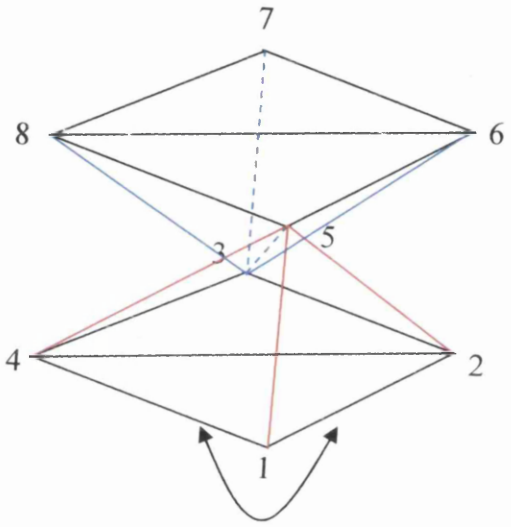
(a) Two planes.



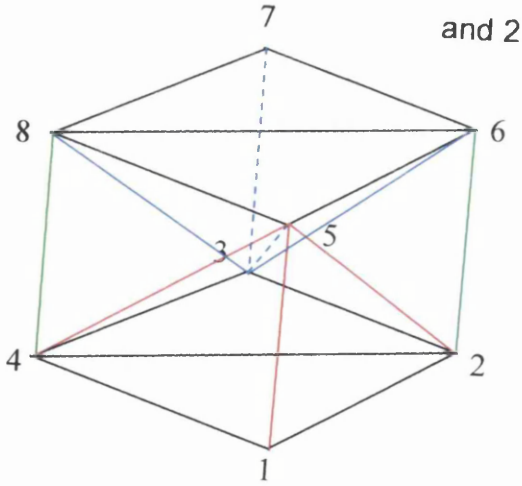
(b) T_1 elements



(c) T_1 and T_2 elements.



(d) External angle between nodes 4 and 2 of the edge 1-5.



(e) Solid domain.

Figure 5.13 Creation of solid geometry.

5.5.3 Elimination of External Elements and Non-Solids

To achieve the required surface mesh, all elements that lie outside of the contours have to be removed. This means that an edge in P_1 or P_2 that lies outside of the contours is removed along with its corresponding tetrahedra, see Figure 5.14. This process is the same as removing all points that lie on the External Voronoi Skeleton.

After this process there may still remain *non-solid connections*, which are tetrahedra that are only connected along an edge or at one single point to one of the two planes, see Figure 5.15. These elements are removed by looking at each vertex and checking for a nonsolid connection.

If these non-solid elements are not removed then a mesh such as that shown in Figure 5.16a will be obtained. The way in which these elements are removed is by checking for solid connections. These can be seen in Figure 5.17 where two solid cubes are connected by an element, which is a non-solid connection. This element is removed by creating a list of every element connected to each node. Then for every node a face connected to this node is obtained, that is either a T_1 or T_2 connection comprising of 3 points in plane 1 or 2.

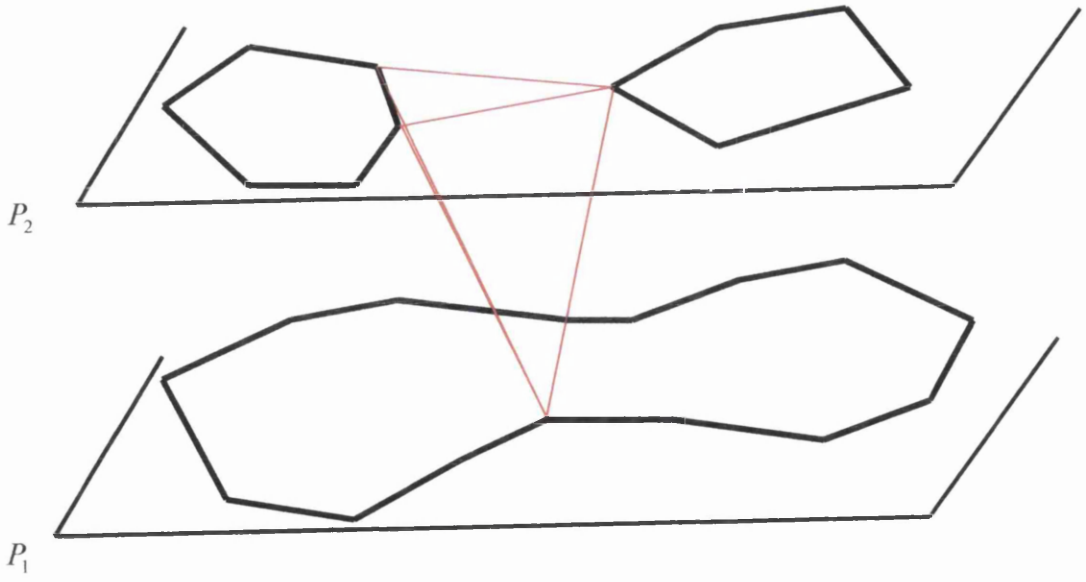


Figure 5.14 Tetrahedron lying outside the contours.

This element is looked at and all elements that are connected to this element via a face are flagged. This is continued for those elements connected to the first element and so on until all elements have been checked. The elements that are not flagged as being connected are removed as they are believed to be a non-solid connection. The resulting mesh of Figure 5.16a can be seen in Figure 5.16b.

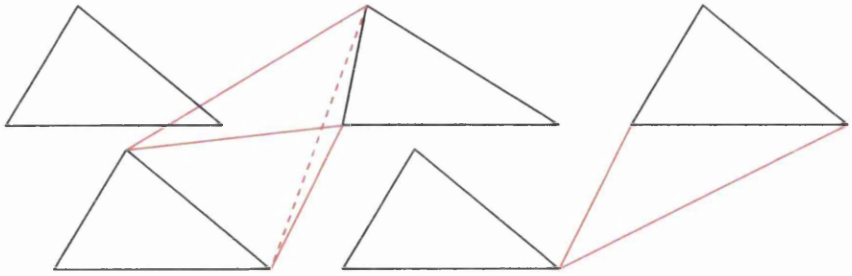
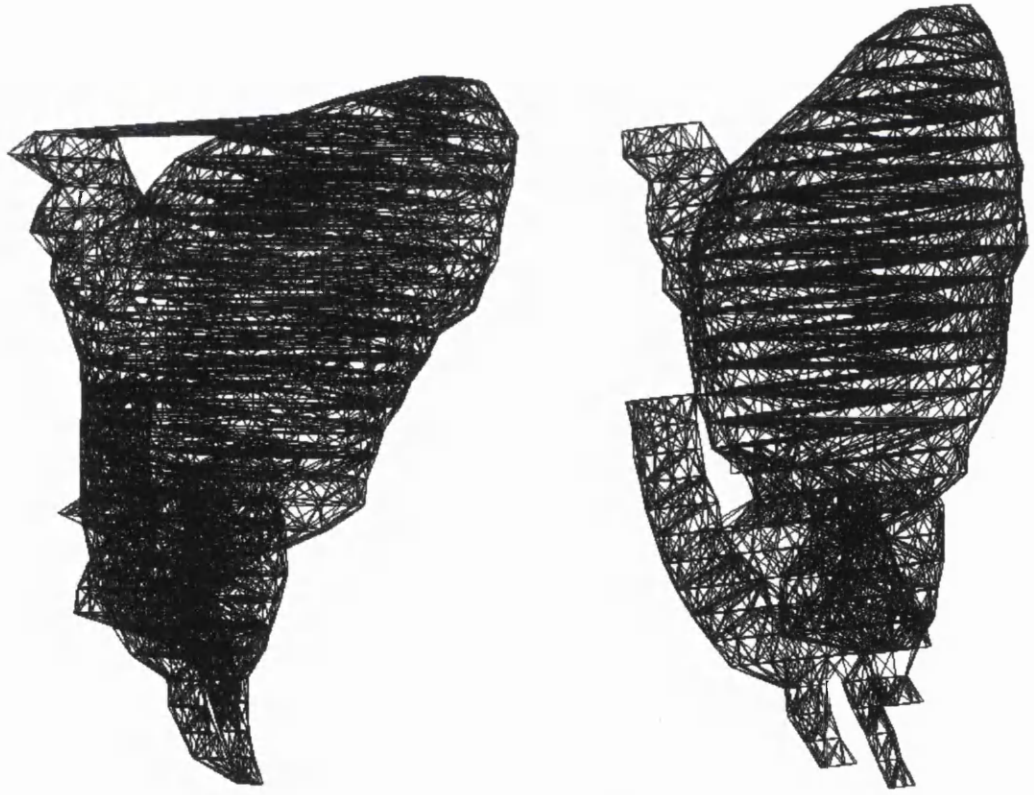


Figure 5.15 Non-solid connections.



(a) Non-solid connections. (b) Non-solids removed.

Figure 5.16 Showing non-solid connection on the heart.

After this process a 3D volume mesh is obtained, where in order to obtain the final 3-D surface mesh the boundary faces of the elements needs to be extracted.

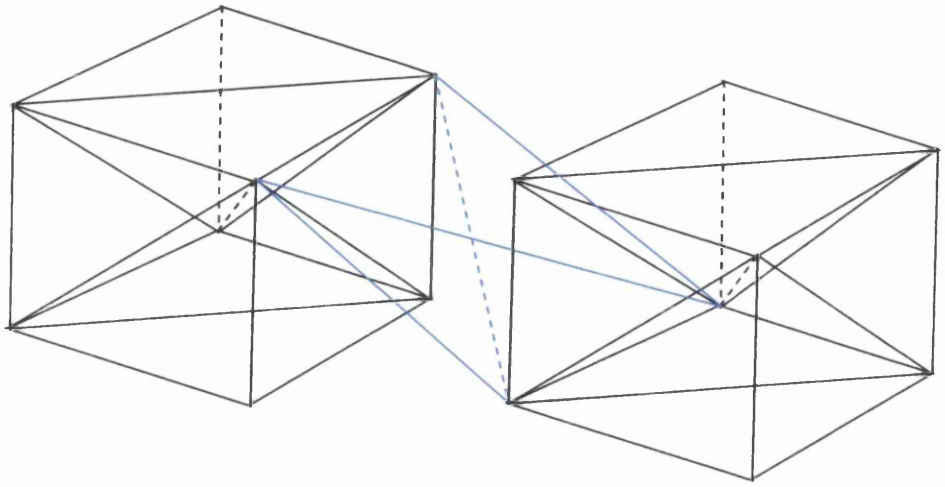


Figure 5.17 Two cubes joined via a non-solid connection.

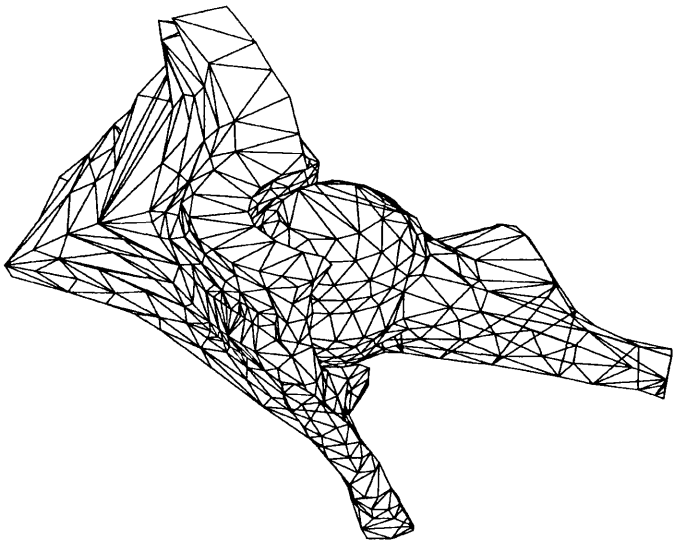
5.6 Examples

Throughout this section a number of different examples will be shown utilising the method described in this chapter. The examples are all related to medical imagery, but the method could also be used for any scanned image.

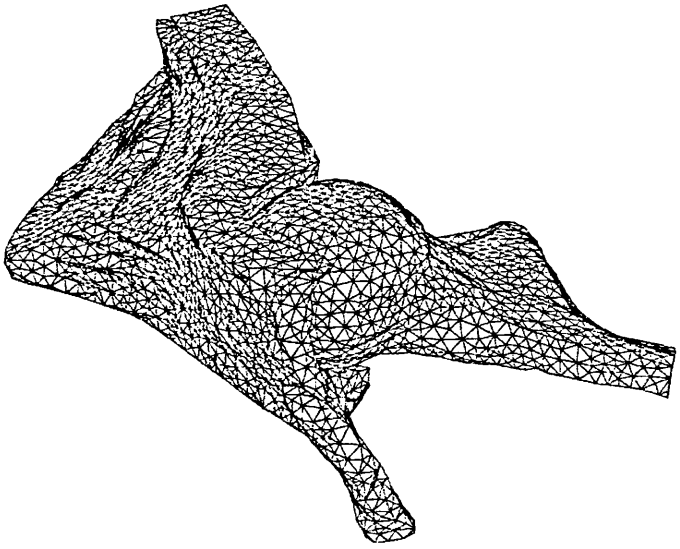
During this project the idea of mesh improvement has been looked at in depth, and this chapter has been used to develop a number of different types of meshes in order to look at mesh improvement. That is why throughout this section the meshes will be checked for quality and then using the method introduced in Chapter 4 for arbitrary geometries, the domain will be re-meshed and checked for an improvement in mesh quality. The purpose of this is to be able to generate meshes from a scanned image and then produce a high quality surface mesh for analysis.

The first example is that of the hip joint. Figure 5.18a shows the geometry created from ten cross sections, 719 nodes and 1432 elements. It can be seen that the mesh is not of a very high quality as shown in Table 5.1 and Table 5.2. Figure 5.18b shows the geometry after the re-meshing process, where it contains 4987 nodes and 9994 elements. The quality of the mesh

has been greatly improved and hence a better mesh for analysis has been produced.



(a) Original mesh.



(b) Re-meshed geometry.

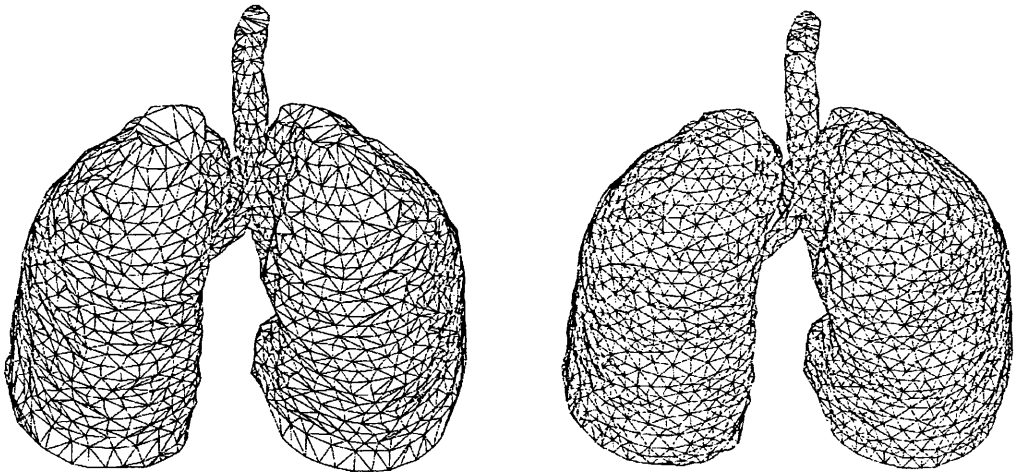
Figure 5.18 Example of hip joint.

The next example is that of the lungs, Figure 5.19a. The geometry is made up of 32 cross sections, 2861 nodes and 5774 elements. It can be seen from Figure 5.19c that there are a number of badly shaped elements. This has been greatly improved once the re-meshing process has been applied, as shown in Figure 5.19b and Figure 5.19d. The resulting mesh

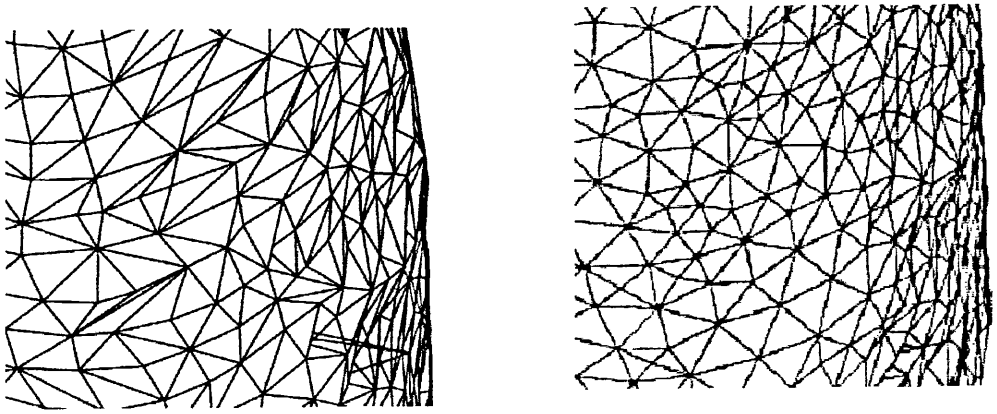
contains 5132 nodes and 10452 elements. The improvement in quality can be seen from Table 5.1 and Table 5.2.

The heart is shown in Figure 5.20a, where once again the problems of the previous example apply. The original mesh contained 29 cross sections, 1209 nodes and 2424 elements. The improved mesh is shown in Figure 5.20b where the geometry is made up of 7920 nodes and 15846 elements.

The largest example tested was that of the pelvis Figure 21, which contained 41 cross sections, 6571 nodes and 13844 elements. The mesh produced here was of far better quality than any of the previous examples. The reason for this is that a lot more nodes and cross sections were used to produce the mesh, which in turn will generate a higher quality mesh. This can be seen in the quality measures in Table 5.1 and Table 5.2.

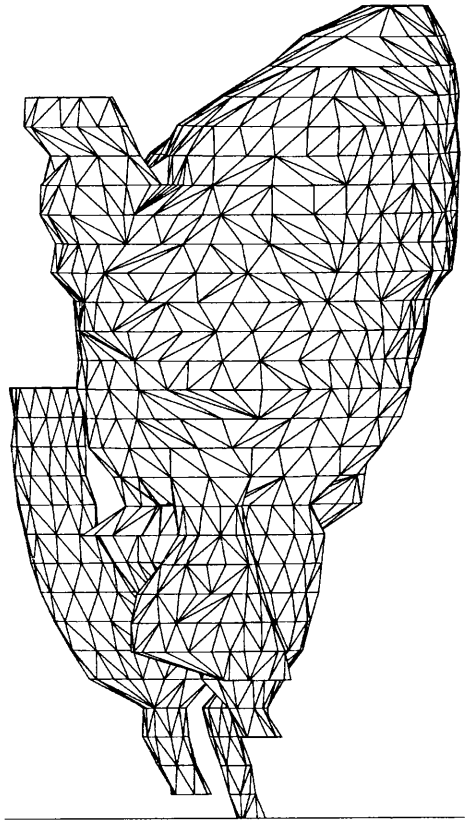


(a) original mesh (b) re-meshed geometry

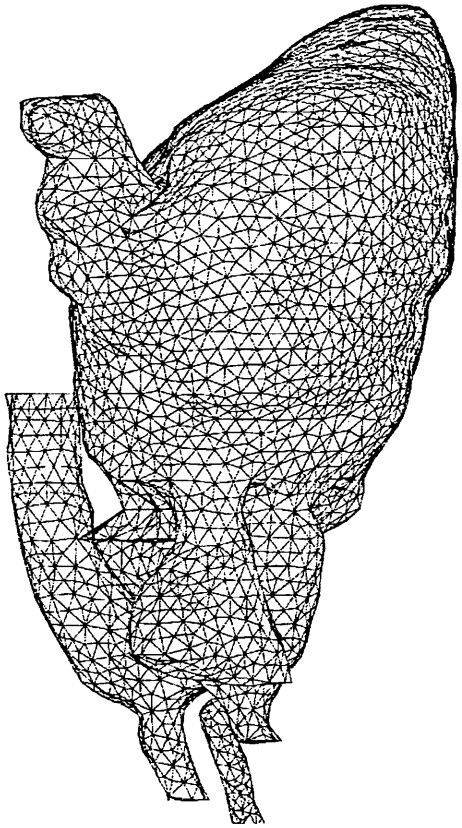


(c) close up of original geometry (d) close up of re-meshed geometry

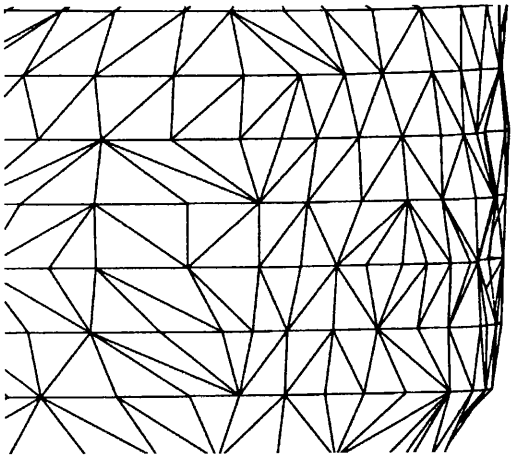
Figure 5.19 Example of lungs.



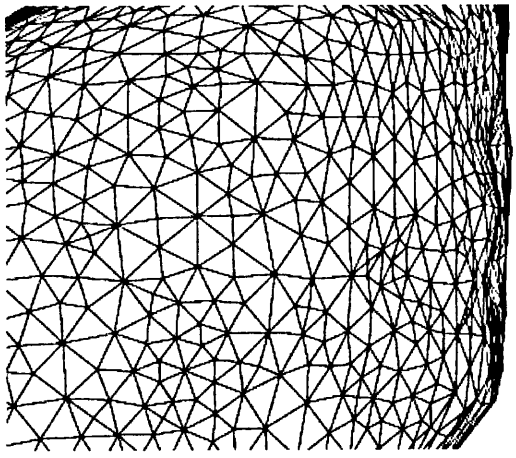
(a) Original mesh.



(b) Re-meshed geometry.



(c) Close up of (a).



(d) Close up of (b).

Figure 5.20 Example of heart.

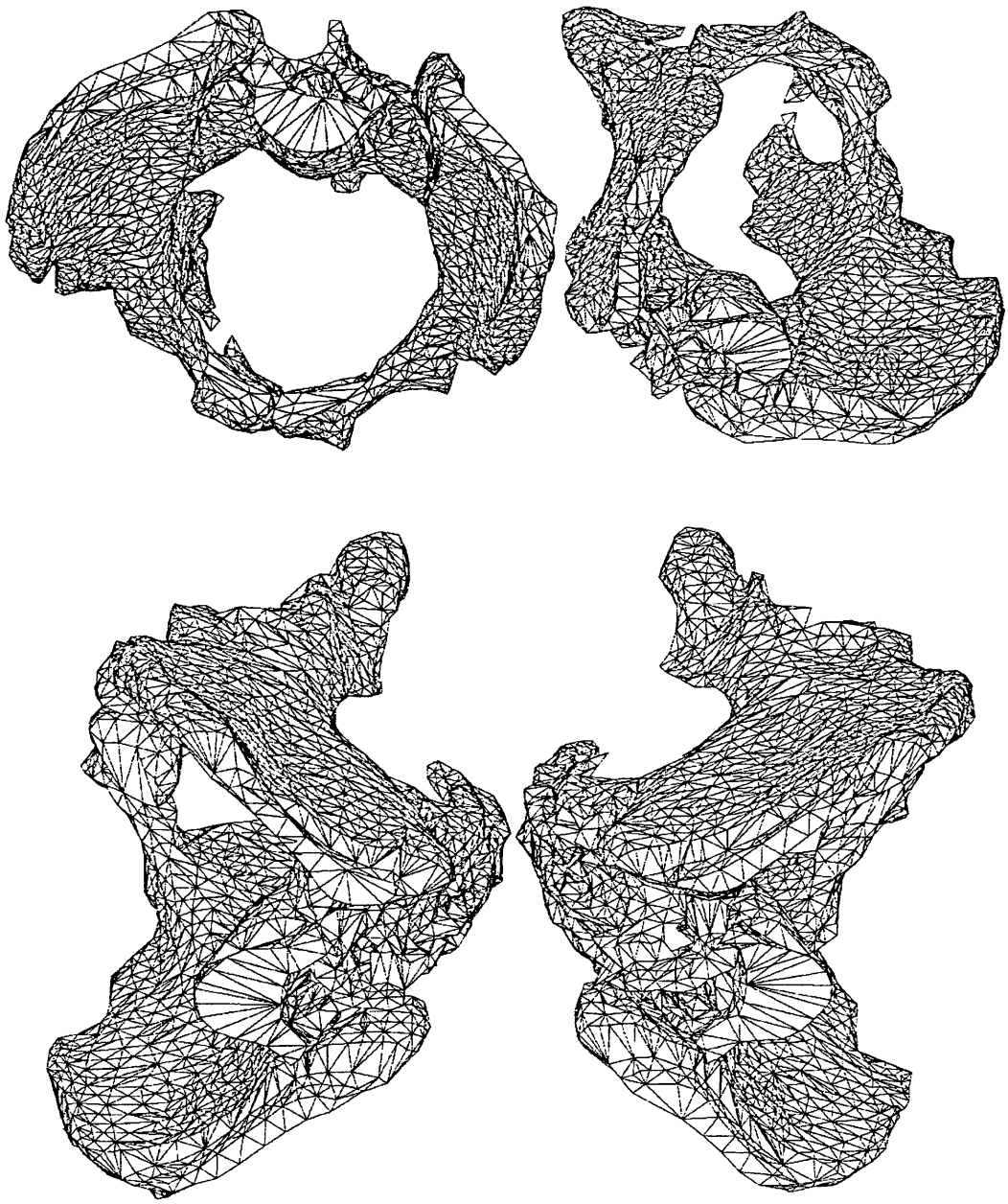


Figure 5.21 Example of the pelvis showing four different projections.

Example	$Qual_{avg}$	$Qual_{G}$	$Qual_{per}$	No elements	No. points
Hip	3.23	7.69	76.21	1432	719
Hip new	1.47	3.10	91.30	4987	9994
Lungs	2.98	9.21	81.10	5774	2861
Lungs new	1.39	4.15	93.22	10452	5132
Heart	2.54	7.12	72.10	2424	1209
Heart new	1.24	3.54	95.7	15846	7920
Pelvis	1.89	6.21	91.23	13844	6571

Table 5.1 Quality measures.

Example	$l_{\min}$	$l_{\max}$	l_{ratio}	$area_{ratio}$	$ang_{\min}$	$ang_{\max}$
Hip	4.539	22.878	5.045	4.930	12.15	107.68
Hip new	1.827	2.129	1.160	1.239	45.71	68.88
Lungs	5.214	17.132	3.286	7.954	8.98	151.14
Lungs new	3.011	3.542	1.177	1.356	41.21	72.43
Heart	10.11	79.43	7.857	2.318	15.17	110.76
Heart new	4.116	4.923	1.196	1.112	43.41	69.98

Table 5.2 Quality measures, length, area and angles.

6.

Concluding Remarks

6.1 Conclusions

Mesh improvement techniques have been presented throughout this thesis. These include two methods of CAD geometry improvement and also the creation of a 3D image from a set of scanned images.

The first method introduced in Chapter 3, was that of the Super Patch method. This method is fully automatic in the choosing and merging of patches in order to obtain a Super Patch. The aim is to merge together neighboring patches of similar tangency in order to create a Super Patch by comparing the angle between the normals of each surface sharing a curve. This can be seen as a limitation of the method as some geometries may have a different tolerance, but this angle can be changed by the user if they wish. The user could also look at curvature instead of angles.

The method introduced in Chapter 3 relies on trimmed patches, it can not deal with overlapping patches although this could be an area of future work. Also the method does not take into account gaps in the geometry, a watertight geometry is assumed. This is an area that could be introduced into the method, by setting a tolerance to define which gaps can be closed. This can be set by the user at the start of the process. The tolerance will state which curves can be merged and would be a geometry dependent process.

In Chapter 4 the method of metric controlled meshing for mesh improvement was introduced. The purpose of this chapter was to introduce a more general approach to mesh improvement techniques that could be used on cases where all that is available is the surface triangulation. This method involves a two-stage process, firstly to obtain a mesh for improvement, secondly to re-mesh the geometry based on a prescribed metric.

Once again the method involves using angles to define ridges. This is something that can vary from geometry to geometry and sometimes even

throughout one geometry, which is why care and knowledge have to be used when defining the angle.

If the mesh that is given is extremely coarse at the start, then this can cause a problem when generating the mesh, as when defining the angle used to declare ridges, the code may select a coarse element as a ridge when in reality it is not.

The computation of the normal at a point used is a weighted average of the surrounding element areas. This too could cause a problem as when computing the normal a badly shaped element may be too influential over the final result.

In order to enhance the capabilities of this method the process of using the surface definition in order to place points back onto the surface could be introduced. This will result in two applications, the G_1 for surfaces with no surface definition and the ability to utilise the surface information if the surface definition is available. Also for this process the curvature could be computed via the original surface geometry, if it is available.

The final method introduced in this thesis was that of surface meshing for scanned images. The method is a fast and efficient way of creating a surface image from a set of scanned data. The quality of the final mesh relies on the accuracy of the nodes placed in the scanned images. This is not something that was looked at in this thesis, as the aim was to use the scanned images to produce the mesh. The improvement of the creation of the scanned images is an area that could be looked at in the future.

The method relies heavily on the contour containment rule being satisfied which is why points sometimes have to be inserted in order to obtain the correct containment, where a Delaunay edge is a boundary edge. This was achieved by placing a point along the edge that needs to be contained. This is not a foolproof method, but for all the cases that were looked at, no problems have occurred.

References

[1] G. E. Farin, 'Curves and surfaces for computer aided geometric design', 1994.

[2] J. C. Ferguson, 'Multivariate Curve Interpolation', Journal ACM, 11, no 2, 221-228, 1964

[3] J. Woodwark, 'Computing Shape', Butterworths, (1986)

[4] A. Rockwood and P. Chambers, 'interactive Curves and Surfaces'. A Tutorial, 1996

[5] Les Piegl and Wayne Tiller, 'The NURBS Book', Springer, (1995)

[6] Pascal Frey, 'about surface re-meshing' proc 9th Int. meshing roundtable, new Orleans, Oct., 2000

[7] G. M. Nielson, 'the side-vertex method for interpolation in triangles', J Approx. theory, 25, 318-336, 1979.

[8] Houman Borouchaki, P. Frey, 'mesh gradation control', J. Numerical methods in eng., 43, 1143-1165, 1998

[9] O. Hassan, N. P. Weatherill, J. Peraire, 'Generation and adaption of unstructured meshes', internal paper university of wales Swansea, CR/872/95, 1995

[10] I. D Faux and M. J Pratt, 'Computational Geometry for design and Manufacture', Ellis Horwood, Chichester, 1981.

[11] N.P. Weatherill, M.J. Marchant, O. Hassan and D.L. Marcum, 'grid adaptation using a distribution of sources applied to inviscid compressible flow simulations', J. numerical methods in fluids, 19, p739-764, 1994

[12] J.J. Stoker, 'differential geometry', Wiley-interscience, 1969

[13] T.J. Willmore, 'An Introduction to differential Geometry', Oxford Press, 1959

[14] Izu Vaisman, 'A First Course in Differential Geometry', Dekker, (1984)

[15] P. Lancaster and K. Salkauskas, 'Curve and Surface Fitting an Introduction', Academic Press, 1986

[16] P. J. Frey and P.L. George, 'Mesh Generation Application to Finite Elements', hermes science, 2000

[17] E. Kreyszig, 'Advanced Engineering Mathematics, seventh edition', Wiley, 1993

[18] P.J. Frey and H. Borouchaki, 'Geometric Evaluation of Finite Element Surface Meshes', J. Finite Elements in analysis and design, 31, p33-53, 1998.

[19] R.A. Adams, 'Calculus a complete course', Addison-Wesley, 1999.

[20] P. J. Frey and H. Borouchaki, 'surface mesh evaluation', int. J. Numerical methods Engineering, 45, p 101-118, 1998

[21] P. Frey, 'Anisotropic surface remeshing', computational fluid and solid mechanics, p1569-1571, 2001

[22] T. J. Baker, 'pruning of surface triangulations based on local curvature', p17-26, 1996

[23] J. F. Thompson, B. K. Soni and N. P. Weatherill, 'handbook of grid generation', CRC, 1999

[24] M.R. Jones, M.A. Price and G. Butlin, 'Geometry Management Support for Auto-meshing', Proc 4th international meshing roundtable, 1995, p153-164.

[25] G. Butlin and C. Stops, 'CAD Data repair', Proc 5th international meshing roundtable, 1996, p7-12.

[26] A. Sheffer, 'Model simplification for meshing using face clustering', 'computer –aided design, 2001, 33:925-934.

[27] A. Sheffer, T. B;acker and M. Bercovier, 'virtual topology operations for meshing', international journal of computational geometry and applications, 2000; 10(3): 309-331.

[28] K.Y. Lee, C.G. Armstrong and M.A. Price, 'orchestrating model simplification prior to finite element mesh generation', 11th ACME conference, 2003, p 97-100.

[29] A. Bowyer, 'Computing Dirichlet tessellations', The Computer Journal, Vol. 24, No. 2, p162-166, 1981.

[30] D. F. Watson, 'Computing the n-dimensional Delaunay Tessellation with Application to Voronoi Polytopes', Computer Journal, Vol. 24, No. 2, p 167-172, 1981.

[31] N. P. Weatherill, O. Hassan, K. Morgan, 'Unstructured grid generation by Delaunay triangulation.', von Karman institute for fluid Dynamics, lecture series 2000-05, March 20-24, 2000.

[32] P. L. George, F. Hecht and M. G. Vallet, 'Creation of internal points in Voronoi type method. Control adaptation', Advances in engineering software p 135-6, 1991.

[33] T. J. Barth, N. L. Wiltberger and A. S. Gandhi, 'Three dimensional unstructured grid generation via incremental insertion and local optimization', Proceddings of software systems for surface modelling and grid generation. Ed. R. E. Smith, NASA Langley, NASA conference proceedings 3143: 449461, 1992.

[34] M. Kobayashi, H. Maekawa and Y. Kondou, 'Automatic discretisation of a three dimensional domain into Voronoi polyhedron elements', JSME international journal. Series H 35.2, 1992.

[35] W. H. Frey, 'Selective refinement: a new strategy for automatic node placement in graded triangular meshes', Int. J. Num. Methods in Eng., 2183-2200, 1987.

[36] T. H. Cormen, C. E. Leiserson and R. L. Rivest, 'introduction to algorithms', the MIT press, 1992

[37] R. Fenn, 'Geometry', Springer, 2001

[38] L. Bostock and S. Chandler, 'Applied mathematics 2', Stanley Thornes Ltd. 1976

[39] L. Bostock and S. Chandler, 'Applied mathematics 1', Stanley Thornes Ltd. 1975

[40] T. M. R. Ellis, 'Fortran 77 programming second edition' Addison-Wesley, 1990

[41] C. F. Gerald and P. O. Wheatly, 'Applied numerical analysis sixth edition', Addison-Wesley, 1999

[42] L. Bostock and S. Chandler, 'Pure Mathematics 1', Stanley Thornes Ltd. 1978

[43] L. Bostock and S. Chandler, 'Pure Mathematics 2', Stanley Thornes Ltd. 1979

[44] G. James, 'Modern engineering mathematics second edition', Addison-Wesley, 1996

[45] M. Spivak, 'Calculus', Addison-Wesley, 1988

[46] B. K. Soni, J. F. Thompson, J. Hauser and P. Eiseman, '5th international conference on numerical grid generation in computational field simulations', April 1 – April 5, 1996

[47] J. R. Sack and J. Urrutia, 'Handbook of computational geometry', North-Holland, 2000

[48] H. Borouchaki, F. Hecht and P. J. Frey, 'Mesh gradation control', International J. for Numerical methods in engineering, 43, 1143-1165, 1998.

[49] E. Keppel, 'Approximating complex surfaces by triangulation of contour lines.', IBM Journal of res. And development, 19: 2-11, 1975.

[50] H. Fuchs, Z. Kedem and S. P. Uselton, 'Optimal surface reconstruction from planer contours.', communications of the ACM, 20: 693-702, 1977.

[51] J. D. Boissonnat, 'Shape reconstruction from plane cross-sections.', computer vision, graphics and image processing, 44: 1-29, 1988.

[52] J. D Boissonnat and B. Geiger, 'Three dimensional reconstruction of complex shapes based on the Delaunay triangulation.', INRIA, rapports de recherché, no. 1697, 1992.

[53] C. Barrillot, B.Gilbaud, L. M. Lou and J. M. Scarabin, 'Three dimensional representation of anatomic structures from CT examination.', in biostereometrics, p 307-314, 1985.

[54] H. N. Christian and T. W. Sederberg, 'Conversion of complex contour line definitions into polygonal element mosaics.', Computer graphics, 8: 658-660, 1978.

[55] K. H. Hoehne and R. Berstein, 'Shading 3-D images from CT scans using gray-level gradients. IEEE Trans. Medical imaging, p45-47, 1986.

[56] D. G. Kirkpatrick, 'Efficient computation of continuous skeletons.', In IEEE symposium on foundations of computer science, 20: p 18-27, 1979.

[57] W. Lorenson and H. Cline, 'Marching cubes, a high resolution 3D surface construction algorithm.', Computer graphics, 21: 163-169, 1987

[58] M. Levoy, 'Display of surfaces from volume data.', IEEE computer graphics and applications, p29-37, 1988.

[59] D. Meyers, S. Skinner and K. R. Sloan, 'Surface from contour: the correspondence and branching problems.', In graphics interface, 1991.

[60] F. P. Preparata and M. I. Shamos, 'Computational Geometry: an introduction.', Springer-Verlag, 1985.

[61] M. R. Stytz, G. Frieder and O. Frieder, '3-D medical imaging: algorithms and computer systems.', ACM computing surveys, 23(4): 421-499, 1991.

[62] M. Shantz,' Surface definition for branching contour defined objects.', Computer graphics, 15: p242-270, 1981.

[63] K. D. Toennies, 'Surface triangulation by linear interpolation in intersection planes.', In science and engineering of medical imaging, p98-105, 1989.

[64] A. J. George, 'computer implementation of the finite element method.', PhD. Thesis, Stanford university, STAN-CS-71-208, 1971

[65] F. Thomasset, 'Implementation of finite element methods for Navier-stokes equations.' ,Springer series in comp. Physics, 1981.

[66] S. H. Lo, 'A new mesh generation scheme for arbitrary planer domains.', int. J. Num. Meth. Eng., 21: p 1403-1426, 1985.

[67] J. Peraire, M. Vahdati, K. Morgan and O. C. Zienkiewicz, 'Adaptive remeshing for compressible flow computations.', J. Comp. Phys., 72: p 449-466, 1987.

[68] S. H. Lo, 'Volume discretisation into tetrahedra-II. 3D triangulation by Advancing Front approach.', Comp. Struct., 39: No 5, p 501-511, 1991.

[69] R. Lohner and P. Parikh, 'Generation of three dimensional unstructured grids by the Advancing Front method.', AIAA Paper AIAA-88-0515, 1988.

[70] J. Peraire, J. Peiro, L. Formaggia, K. Morgan and O. C. Zienkiewicz, 'Finite element euler computations in three dimensions.', Int. J. Num. Meth. Eng., 26: p 2135-2159, 1988.

[71] J. Peraire, J. Peiro and K. Morgan, ' Adaptive remeshing for three dimensional compressible flow computations.', J. Comp. Phys., 103: p 269-285, 1992.

[72] L. Formaggia, 'An unstructured mesh generation algorithm for three dimensional aircraft configurations.', Numerical grid generation in CFD and related fields. (Ed.) Sanchez-Arcilla, north Holland, p 249-259, 1991.

[73] J. Frykestig, 'Advancing front mesh generation techniques with application to the finite element method.', Dept. of structural mechanics publication 94, 10, Chalmers university of technology, Goteborg, Sweden, 1994.

[74] H. Jin and R. I. Tanner, 'Generation of unstructured tetrahedral meshes by Advancing Front technique.', Int. J. Num. Meth. Eng. 36: p 1805-1823, 1993.

[75] R. Lohner, 'Extensions and improvements of the Advancing Front grid generation technique.', Comm. Num. Meth. Eng., 12:p 683-702, 1996.

[76] P.Moller and P. Hansbo, 'On Advancing Front mesh generation in three dimensions.', Int. J. Num. Meth. Eng., 38:p 3551-3569, 1995.

[77] J. Z. Zhu, O.C. Zienkiewicz, E. Hinton and J.Wu, 'A new approach to the development of automatic quadrilateral mesh generation.', Int. J. Num. Meth. Eng., 32:p 894-866, 1991.

[78] T. D. Blacker and M. B. Stephenson, ' Paving a new approach to automated quadrilateral mesh generation.', Int. J. Num. Meth. Eng., 32: p 811-847, 1991.

[79] T. D. Blacker and R. J. Meyers, 'Seams and wedges in plastering: a 3D hexahedral mesh generation algorithm.', Eng computers, 9: p 83-93, 1993.

[80] J. F. Thompson, 'A reflection on grid generation in the 90's: trends, needs and influences.', NSF engineering research centre for computational field simulation, 2001.

[81] W. J. Gordon and R. F. Riesenfeld, 'B-splines curves and surfaces.', in computer aided geometric design, Academic Press, New York, p95-126, 1974.

[82] P. Bezier, 'Numerical Control: mathematics and its applications.', John Wiley and sons, London, 1972.

[83] D. Attali and A. Montanvert. 'Computing and simplifying 2D and 3D continuous Skeletons.', computer vision and image understanding, 67(3) p 261-273, 1997.

[84] T. Culver, J. Keyser and D. Manocha, ' Accurate computation of the medial axis of a polyhedron.', UNC technical report TR98-038. ACM solid modelling, 1999.

[85] D. Sheehy, C. Armstrong and D. Robinson, 'shape description by medial axis construction.', IEEE transactions on visualization and computer graphics, 2(1), p 62-72, 1996.

[86] C. Armstrong, D. J. Robinson, R. M. McKeag, T. S. Li, S. J. Bridgett, R.J. Donaghy and C. A McGleenan, 'medials for meshing and more.', Quenns university Belfast (http://sog1.me.qub.ac.uk/femgroup.html).

[87] P. Yang Ang and C. Armstrong, 'Adaptive curvature-sensitive meshing of the medial axis.', Queens university Belfast (http://sog1.me.qub.ac.uk/femgroup.html).

[88] E. Artzy, G. Frieder and G. T. Herman, 'The theory, design, implementation and evaluation of a 3D surface detection algorithm.', Comput. Graphics image process. 15, p 1-24, 1981.

[89] R. Ribo, G. Bugeda and E Onate, 'Some algorithms to correct a geometry in order to create a finite element mesh.' , Comput. Aided Design 80:1399-1408, 2002.

[90] D. L. Marcum and J. A. Gaither, 'Unstructured Surface Grid Generation Using Global Mapping and Physical Space Approximation.' *Proceedings, 8th International Meshing Roundtable*, South Lake Tahoe, CA, U.S.A., pp.397-406, October 1999

[91] G. L. Dirichlet, 'Uber die reduction der positiven Quadratischen formen mit drei underestimrnten ganzen zahlen', Reine Angew. Math., 40, no.3 p 209-227, 1850.

[92] T. J. Baker, 'Three-Dimensional mesh generation by triangulation of arbitrary point sets.' ,Proceedings of the AIAA 8th CFD conference, Hawaii, June 1987.

[93] G. Farin, 'surfaces over Dirichlet tessellations.' Compuer aided geometric design, 7 p 281, 1990.

[94] J. C. Cavendish, D. A. Field and W. H. Frey, 'An approach to automatic three-dimensional finite element mesh generation.' Meth. Eng. 21, p329, 1985.

[95] D. N. Shenton and Z. J. Cendes, 'three-dimensional finite element mesh generation using Delaunay tessellation.' IEEE trans. Magnetics, MAG-21, p 2535, 1985.

[96] P. L. George, F. Hecht and E. Saltel, 'Tetraedrisation automatique et respect de la frontiere.' INRIA Report 835, 1988.

[97] P. L. George, F. Hecht and E. Saltel, 'constraint of the boundary and automatic mesh generation.' Proc. 2nd international conference on numerical grid generation in computational fluid mechanics, sengupta, s., (Ed.), pineridge press, p589, 1988.

[98] A. Perronnet, 'Un algorithme de tetraedrisation d'un object multi-materiaux ou de L'exterieur d'un object.' Numerical analysis laboratory report. (R88005). University Pierre et marie curie, 1988.

[99] W.J. Schroeder and M. S. Shephard, 'Geometry-based fully automatic mesh generation and the Delaunay triangulation.' Int. j. num. Meth. Eng. 26, p2503, 1988.

[100] D. Sharov and K. Nakahashi, 'A boundary recovery algorithm for Delaunay tetrahedral meshing.' Proc. 5th international conference on numerical grid generation in computational field simulations, Soni, B.K. and Thompson, J.F., (Ed.) NSF Engineering research center for computational field simulations p229, 1996.

[101] W. F. Kern and J. R. Bland, 'Solid mensuration with proofs, 2nd edition.' New York. Wiley. 1948.

[102] D. E. Knuth, 'The art of computer programming', Vol 1, Fundamental Algorithms, Addison Wesley, Reading, Ma, 1969.

[103] R. Sedewick, 'algorithms', 2nd Edition, Addison Wesley, Reading, Ma, 1988.

[104] J. Bonet and J. Peraire, 'An Alternate Digital Tree algorithm for geometric searching and intersection problems', University College of Swansea report C/R/619/88, 1988.

[105] N. P. Weatherill, 'Generation of unstructured grids using Dirichlet tessellations', Princeton University, 1985.

[106] Georgy Voronoi, 'Nouvelles applications des paramètres continus à la théorie des formes quadratiques.' Journal fur die Reine und Angewandte Mathematik, 133:97-178, 1908.

[107] B. Delaunay, Sur la sphère vide, Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk, 7:793-800, 1934

[108] "Automatic 3D Mesh Generation Conforming a Prescribed Size Map", Numerical Grid Generation in Computational Field Simulations, Ed. M. Cross., B. K. Soni, J. F. Thompson, J. Hauser, P. R. Eiseman, Proceedings of the 6th International Conference, held at the University of Greenwich, pp.21-32, July 1998.

[109] D-Lib magazine article entitled "Accessing the Visible Human Project®" by Michael J. Ackerman, Ph.D.

[110] NLM's Current Bibliographies in Medicine, Visible Human Project (CBM 2000-5): 425 citations from January 1987 - August 2000.

[111] Banvard, Richard A., The Visible Human Project® Image Data Set From Inception to Completion and Beyond, Proceedings CODATA 2002: Frontiers of Scientific and Technical Data,

Track I-D-2: Medical and Health Data, Montréal, Canada, October, 2002.

[112] C. L. Bajaj, K. Lin, and E. Coyle. Arbitrary Topology Shape Reconstruction from Planar Cross Sections. *Graphical Models and Image Processing*, 58(6):524-543, 1996.